Sistema de Telemetría basado en redes WSN (Wireless Sensor Network) para el Internet de las Cosas (IoT)

Autor: Dixys Leonardo Hernández Rojas

Tesis Doctoral UDC / 2018

Tutor: Carlos Escudero Cascón

Programa de Doctorado en Tecnologías da Información
y Comunicaciones en Redes Móviles

UNIVERSIDADE DA CORUÑA

12 Septiembre, 2018
Universidade da Coruña
Facultad de Informática
Campus de Elviña s/n
15071 - A Coruña (España)

Aviso de Copyright:
Todos los derechos reservados. Ninguna parte de esta publicación puede reproducirse, almacenarse en un sistema de recuperación o transmitirse de ninguna forma ni por ningún medio (electrónico, mecánico, fotocopia, grabación u otro) sin la previa autorización escrita del autor.

Dr. Carlos Escudero Cascón

CERTIFICA

Que la memoria titulada:
"Sistema de Telemetría basado en redes WSN (Wireless Sensor Network) para el Internet de las Cosas (IoT)"
fue realizada por el Ing. Dixys Leonardo Hernández Rojas bajo mi dirección en el Departamento de Ingeniería de Computadores desde Abril del 2014 en la Universidade da Coruña y finaliza la Tesis que presenta para optar el grado de Doctor.

A Coruña, 12 de Septiembre del 2018.

Asdo.: Dr. Carlos Escudero Cascón
Director de la Tesis Doctoral
Titular de Universidade
Departamento de Ingeniería de Computadores
Universidade da Coruña

Dixys Leonardo Hernández Rojas
Doctorando

Tesis doctoral: Sistema de Telemetría basado en redes WSN (Wireless Sensor Network) para el Internet de las Cosas (IoT)

Autor: Ing. Dixys Leonardo Hernández Rojas

Director: Dr. Carlos Escudero Cascón

Datos de la defensa:

Tribunal

Presidente:

Vocal:

Secretario:

Agradecimientos

Esta tesis doctoral, no solo ha sido una satisfacción personal, con ella queremos pretender dar ejemplo a nuestros hijos, alumnos y amigos en la posibilidad de realizar investigación científica como forma de vida y no como la culminación de un ciclo de estudio. El desarrollo tecnológico y la investigación en sí ha sido siempre nuestra pasión y ha llevado a momentos difíciles de superación, hemos tenido que sacrificar sobre todo el tiempo para mi familia, así que este logro es también de ellos.

Un agradecimiento para el Dr. Luis Castedo, quien nos permitió acceder al programa doctoral que dirige y ser parte del grupo de investigación GTEC de la Universidad de la Coruña. Un agradecimiento muy especial a mi tutor y director Dr. Carlos Escudero Cascón. Su toma de decisión y orientación oportuna en las diferentes fases del desarrollo de nuestro trabajo, marcan sin duda una guía a seguir en el futuro de nuestro desempeño profesional y en el arduo y a veces ingrato camino de la investigación científica.

Mis sinceros agradecimientos a todos los colegas, profesores y excelentes asesores del grupo GTEC, en especial a Tiago Fernández Caramés y Paula Fraga Lamas, los cuáles de forma desinteresada han aportado sus conocimientos, apoyo científico y profesional a nuestro trabajo.

Abstract

The Internet of Things (IoT) involves a wide variety of heterogeneous technologies and resource-constrained devices that interact with each other. Due to such constraints, IoT devices usually require lightweight protocols that optimize the use of resources and energy consumption. Among the different commercial IoT devices, Bluetooth and BLE based beacons, which broadcast periodically certain data packets to notify their presence, have experienced a remarkable growth, specially due to their application in indoor positioning systems.

As a solution to this problem, this thesis contributes with a general architecture of telemetry, different scenarios to access the cloud computing, guidelines for the selection of components in the different domains of an IoT-based architecture and examples of implementation of a WSN through a testbed based on the use of smart sensors with Bluetooth Low Energy. In the example shown, an IoT gateway is developed on an Android smartphone. This gateway integrates an MQTT client and has the necessary functionalities to pack the data from the sensors and send them to the MQTT broker residing in a cloud computing. There is also a friendly interface available for a local user to monitor and control the WSN created. Some experiments were carried out to demonstrate the functionality, speed and stability of the proposed gateway, to be used in this and other application domains.

This thesis proposes a family of protocols named LP4S that provides fast responses and enables plug-and-play mechanisms that allow IoT telemetry systems to discover new nodes and to describe and auto-register the sensors and actuators connected to a beacon. Thus, three protocols are defined depending on the beacon hardware characteristics: LP4S-6 (for resource-constraint beacons), LP4S-X (for more powerful beacons) and LP4S-J (for beacons able to run complex firmware). In order to demonstrate the capabilities of the designed protocols, the most restrictive (LP4S-6) is tested after implementing it for a telemetry application in a beacon based on Eddystone (Google's open beacon format). Thus, the beacon specification is extended in order to increase its ability to manage unlimited sensors in a telemetry system without interfering in its normal operation with Eddystone frames. The performed experiments show the

feasibility of the proposed solution and its superiority, in terms of latency and energy consumption, with respect to approaches based on Generic Attribute Profile (GATT) when multiple users connect to a mote or in scenarios where latency is not a restriction, but where low-energy consumption is essential.

This thesis also presents a Virtual Transducer Electronic Data Sheet (VTEDS) based framework for the development of intelligent sensor nodes with plug-and-play capabilities in order to contribute to the evolution of the Internet of Things (IoT) towards the Web of Things (WoT). It makes use of new lightweight protocols that allow the sensors to self-describe, auto-calibrate and auto-register. Such protocols enable the development of novel IoT solutions while guaranteeing low latency, low power consumption and the required QoS. In order to evaluate the performance of the system, it was tested when using Bluetooth Low Energy (BLE) and Ethernet-based smart sensors in different scenarios. Specifically, user experience was quantified empirically (i.e., how fast the system show collected data to a user). The obtained results show that the proposed VTED architecture is really fast, being some smart sensors (located in Europe) able to self-register and self-configure in a remote cloud (in South America) in less than 3 s and to display data to remote users in less than 2 s.

Finally, technological trends for the development of intelligent sensors in the field of the current Internet of things are shown, based on communication technologies and protocols, hardware, multi-protocol and multi-network devices, operating systems, on-line development environments and programming languages of the firmware of said sensor nodes. All the scientific contributions of this thesis are reflected in the design of a precision agriculture project, to control the irrigation of a banana plantation in an automated way. The aforementioned project is still in execution, although progress in its implementation is shown in the last chapter.

Resumen

El Internet de las cosas (IoT) involucra una amplia variedad de tecnologías heterogéneas y dispositivos con recursos limitados que interactúan entre sí. Debido a tales restricciones, los dispositivos de IoT generalmente requieren protocolos livianos que optimicen el uso de los recursos y el consumo de energía. Entre los diferentes dispositivos comerciales de IoT, los beacons basados en Bluetooth y BLE, que transmiten periódicamente ciertos paquetes de datos para notificar su presencia, han experimentado un notable crecimiento, especialmente debido a su aplicación en sistemas de posicionamiento en interiores.

Como solución a esta problemática esta tesis aporta con una arquitectura general de telemetría, diferentes escenarios para acceder a la cloud computing, guías de selección de componentes en los diferentes dominios de una arquitectura basada en IoT y ejemplos de implementación de una WSN a través de un testbed basado en el uso de sensores inteligentes con Bluetooth Low Energy. En el ejemplo mostrado se desarrolla un gateway IoT sobre un teléfono inteligente Android. Este gateway integra un cliente MQTT y dispone de las funcionalidades necesarias para empaquetar los datos de los sensores y enviarlos hacia el broker MQTT residente en una cloud computing. También se dispone de una interfaz amigable para que un usuario local pueda monitorear y controlar la WSN creada. Algunas experimentos fueron llevados a cabo para demostrar la funcionalidad, rapidez y estabilidad del gateway propuesto, para ser usado en este y otros dominios de aplicación.

Como parte de las soluciones planteadas en esta tesis, se propone una familia de protocolos denominada Lightweight Protocol for Sensors (LP4S) que proporciona respuestas rápidas y habilita los mecanismos plug-and-play que permiten a los sistemas de telemetría IoT descubrir nuevos nodos y describir y registrar automáticamente los sensores y actuadores conectados a un beacon. Por lo tanto, se definen tres protocolos dependiendo de las características de hardware del beacon: LP4S-6 (para beacons de recursos restringidos), LP4S-X (para beacons más potentes) y LP4S-J (para beacons capaces de ejecutar firmware complejo). Para demostrar las capacidades de los protocolos diseñados, el más restrictivo (LP4S-6) se prueba después de implementarlo para una aplicación de telemetría en un beacon basado en Eddystone (formato de beacon abierto

de Google). Por lo tanto, la especificación del beacon se amplía para aumentar su capacidad de administrar sensores ilimitados en un sistema de telemetría sin interferir en su funcionamiento normal con las tramas Eddystone. Los experimentos realizados muestran la viabilidad de la solución propuesta y su superioridad, en términos de latencia y consumo de energía, con respecto a los enfoques basados en el perfil genérico de atributos (GATT) cuando múltiples usuarios se conectan a un sensor inteligente o en escenarios donde la latencia no es una restricción, pero el bajo consumo de energía es esencial.

Se presenta además un framework basado en TEDS virtuales (VTEDS) para el desarrollo de nodos de sensores inteligentes con capacidades plug-and-play para contribuir a la evolución del Internet de las Cosas (IoT) hacia la Web de las Cosas (WoT). En el sistema propuesto se utilizan los protocolos livianos, que hemos desarrollado, para que los sensores se auto-describan, auto-calibren y se auto-registren automáticamente. Dichos protocolos permiten el desarrollo de nuevas soluciones de IoT al tiempo que garantizan baja latencia, bajo consumo de energía y la QoS requerida. Para evaluar el rendimiento del sistema, se usaron sensores Bluetooth Low Energy (BLE) y basados en Ethernet en diferentes escenarios. Específicamente, la experiencia del usuario se cuantificó empíricamente (es decir, qué tan rápido el sistema muestra los datos recopilados a un usuario). Los resultados obtenidos muestran que la arquitectura VTED propuesta es realmente rápida, ya que algunos sensores inteligentes (ubicados en Europa) pueden auto-registrarse y auto-configurarse en una cloud remota (en América del Sur) en menos de 3s y mostrar datos de los sensores a usuarios remotos en menos de 2s.

Finalmente, se muestran las tendencias tecnológicas para el desarrollo de sensores inteligentes en el ámbito del Internet de las cosas actual, en función de las tecnologías y protocolos de comunicación, hardware, dispositivos multi-protocolos, sistemas operativos, ambientes de desarrollo on-line y lenguajes de programación del firmware de dichos nodos sensores. Todos los aportes científicos de esta tesis son reflejadas en el diseño de un proyecto de agricultura de precisión, para controlar el riego de una plantación de banano de forma automatizada. El proyecto citado aún se encuentra en ejecución, no obstante los avances de implementación del mismo son mostrados en el último capítulo.

Resumo

A Internet das Cousas (IoT) implica unha ampla variedade de tecnoloxías e dispositivos con recursos limitados interactúan heteroxéneos. Debido a estas condicións, os dispositivos de Internet das cousas en xeral, esixen protocolos leves que optimizan o uso de recursos e consumo de enerxía. Entre os moitos dispositivos comerciais IDC, balizas baseado Bluetooth e ble, que transmiten periodicamente determinados paquetes de datos para notificar a súa presenza, presentan un crecemento significativo, sobre todo debido á súa aplicación en sistemas de posicionamento internos.

Como solución a este problema desta tese ofrece un xeneral arquitectura telemetría escenarios diferentes para acceder a computación en nube, orienta a selección de compoñentes nos diferentes ámbitos de unha economía baseada en Internet das cousas e exemplos de posta en marcha dunha WSN través da arquitectura cun testbed baseado no uso de sensores intelixentes con Bluetooth Low Energy. No exemplo que se mostra, unha pasarela IoT está desenvolvida nun teléfono intelixente con Android. Este portal integra un cliente e ten MQTT necesario para embalaxe os datos dos sensores e envialos para o corrector MQTT residente en capacidades de computación en nube. Hai tamén unha interface amigable dispoñible para que un usuario local monitor e controle o WSN creado. Algunhas experiencias foron realizadas para demostrar a funcionalidade, a velocidade ea estabilidade do porto de conexión proposto, para uso nesta e outros dominios de aplicación.

Como parte das solucións propostas nesta tese, unha familia de protocolos chamados LP4S dando respostas rápidas proposta e permitir mecanismos de plug-and-play que permiten sistemas IDC telemetría descubrir novos nós e describir e automaticamente gravar os sensores e actuadores conectados a un faro. LP4S-6 (por balizas recurso-constrangidos), LP4S-X (por balizas máis fortes) e LP4S-J (por balizas capaces de execución do firmware complexo): Por conseguinte, tres protocolos, dependendo das características de hardware baliza definida . Para demostrar as capacidades de protocolos deseñados, as máis restritivas (LP4S-6) é probada tras a posta en marcha dunha solicitude de Eddystone baseada telemetría (formato aberto faro de Google) Beacon. Polo tanto, o faro especificación expande para aumentar a súa capacidade de xestionar

sensores ilimitadas nun sistema de telemetría sen interferir co funcionamento normal cadros Eddystone. As experiencias mostran a viabilidade da solución proposta ea súa superioridade en canto a consumo de latencia e de potencia en comparación coas propostas baseadas no perfil de atributo xenérico (GATT) cando varios usuarios conectarse a un sensor intelixente ou escenarios onde a latencia non é unha restrición, pero onde o consumo de enerxía é esencial.

baseado Teds virtuais (VTEDS) para o desenvolvemento de nós sensores intelixentes con plug-and-play para contribuír á evolución de Internet das Cousas (Internet das cousas) á Web of Things (WOT) cadro tamén se mostra. Nos protocolos leves sistema proposto, realizamos son usados para que os sensores son propio descrición, auto-calibración e auto-rexistrar automaticamente. Estes protocolos permiten o desenvolvemento de novas solucións IOT asegurando ao mesmo tempo baixa latencia, baixo consumo de enerxía ea QoS requiridos. Para avaliar o rendemento do sistema, eles sensores Bluetooth Low Energy (ble) foron utilizados e Ethernet en base a diferentes escenarios. En concreto, a experiencia do usuario é empiricamente cuantificados (é dicir, o quão rápido o sistema exhibe os datos recollidos a un usuario). Os resultados mostran que a arquitectura VTED proposta é moi rápido, porque algúns sensores intelixentes (situadas en Europa) poden auto-rexistro e auto-configurado nunha nube remota (en América do Sur) en menos de 3s e dos datos de visualización sensores para usuarios remotos en menos de 2 s.

Por último, as tendencias da tecnoloxía para o desenvolvemento de sensores intelixentes no campo de Internet das cousas reais, dependendo das tecnoloxías e protocolos de comunicación, hardware, dispositivos multi-protocolo, sistemas operativos, ambientes de desenvolvemento en liña mostra e linguaxes de programación do firmware dos devanditos nodos de sensores. Todas as contribucións científicas desta tese son reflectidas no deseño dun proxecto de agricultura de precisión, para controlar o risco dunha plantación de bananas automatizado. O devandito proxecto aínda está en execución, a pesar dos progresos da súa execución móstranse no último capítulo.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

El Internet de las Cosas (IoT) constituye un avance importante para la sociedad. Millones de usuarios, hombres y máquinas, participan a nivel mundial activamente en Internet tanto en su vida laboral como en la social y gracias a las tecnologías inalámbricas disponibles, han ampliado sus posibilidades de interacción en la red en cualquier lugar y momento. Por tanto, la tecnología sirve como herramienta de colaboración y toma de decisiones en un mundo en el que converge lo físico con lo digital.

El IoT implica un escenario donde las "cosas", típicamente dispositivos electrónicos inteligentes con sensores y actuadores distribuidos geográficamente, se encuentran identificados y conectados a Internet, que permiten el control y monitoreo remoto de situaciones críticas de un dominio de aplicación, incluso sin la interacción humana. Sin embargo, para poder detectar dichas situaciones es necesario comunicar, almacenar, analizar y procesar eficientemente la gran cantidad de información generada cada día. Una aplicación importante de IoT es la Agricultura Inteligente (Smart Agriculture) y se define como el uso de las Tecnologías de la Información y Comunicaciones en la gestión localizada de cultivos o parcelas agrícolas, basado en la existencia de variabilidad en campo, para aplicar el tratamiento adecuado en el momento justo.

El término Internet de las cosas es una extensión del ya conocido Internet, del cual hoy en día todos somos usuarios permanentes y lo conocemos como la gran autopista de la información, donde podemos encontrar prácticamente cualquier información que necesitamos gracias a la interconexión de millones de computadoras, bases de datos y usuarios alrededor de todo el mundo. Podríamos decir que la evolución del Internet que conocemos hoy en día comenzó en la década de los 70 a los 80, donde se crearon los primeros protocolos de comunicaciones que son la base de nuestro Internet, por las principales universidades de Estado Unidos, como el Instituto Tecnológico de Massachusetts (MIT) y la Universidad de California, Los Ángeles (UCLA), que convirtieron en los 90 a la red militar y académica ARPANET en el Internet actual.

Pero mucho antes, Nikola Tesla (1926) y Alan Turing (1950) anticiparon la conectividad global, la miniaturización tecnológica y la necesidad de inteligencia en sensores y equipos de comunicación, incluso en 1874 científicos franceses lograron transmitir información meteorológica desde la cima del monte Mont Blanc hasta París, constituyendo los primeros experimentos de telemetría hasta ahora registrados. Continuando con la historia, con la aparición del Internet la década de los 90 tuvo una gran actividad de desarrollo tecnológico creando cada vez más aplicaciones pensadas para Internet. Fue así que en 1990 John Romkey, en el evento Interop, mostró al mundo el primer objeto (thing) conectado a Internet, una tostadora que podía ser encendida o apagada remotamente. Luego en 1999 fue creado el centro Auto-ID dentro del MIT, por sus fundadores Sanjay Sarma, David Brock y Kevin Ashton, que lograron enlazar en Internet objetos a través de tarjetas Radio-frequency identification (RFID) . Pero no fue hasta que el director ejecutivo de Auto-ID, Kevin Ashton en este mismo año y luego David L. Brock en el 2001, que acuñaron el término Internet de las Cosas en la historia del Internet.

El concepto del Internet de las Cosas ha tenido múltiples definiciones desde 1999 hasta nuestros días, refiriéndose en sus inicios a solo cosas identificables vía RFID exclusivamente, añadiéndoles inteligencia y mayor ámbito. Podemos decir que el Internet de las cosas actual sería el conjunto de objetos inteligentes, perfectamente auto-identificables, capaces de interactuar remotamente entre sí y con el resto de los equipos conectados a través de Internet en tiempo real, incluso sin la interacción humana. En ciertos dominios de aplicación, como el ámbito agropecuario, IoT lo se ve como el conjunto de sensores capaces de medir parámetros climáticos, suelo, agua, cultivos (tronco, fruto, clorofila, etc.) y otros, que envían la información a un servidor local o remoto (nube); proporcionando a los empresarios agropecuarios y clientes finales la posibilidad de monitorear su producción remotamente desde un terminal conectado a Internet, sea este una PC, Tablet o smartphone.

Actualmente existen varias arquitecturas de IoT y para este capítulo se ha propuesto la arquitectura de la figura 1.1 que ha sido adaptada de [1] y consta de 3 capas: Dominio de Aplicación, Dominio de Red y Dominio de sensores.

En la capa de dominio de Aplicación se encuentra la infraestructura de comunicación, almacenamiento y procesamiento de datos, así como las herramientas de análisis y presentación de la información al usuario. La infraestructura puede estar formada desde un servidor físico o virtualizado a un Centro de Procesamiento de Datos (CPD) complejo que involucra un conjunto de recursos físicos, lógicos y humanos para el control de los procesos y datos en el contexto de IoT. La virtualización de los recursos físicos y disponibilidad en internet se conoce como computación en la nube o Cloud Computing [2–5]. Algunas de las principales funciones de esta capa son:

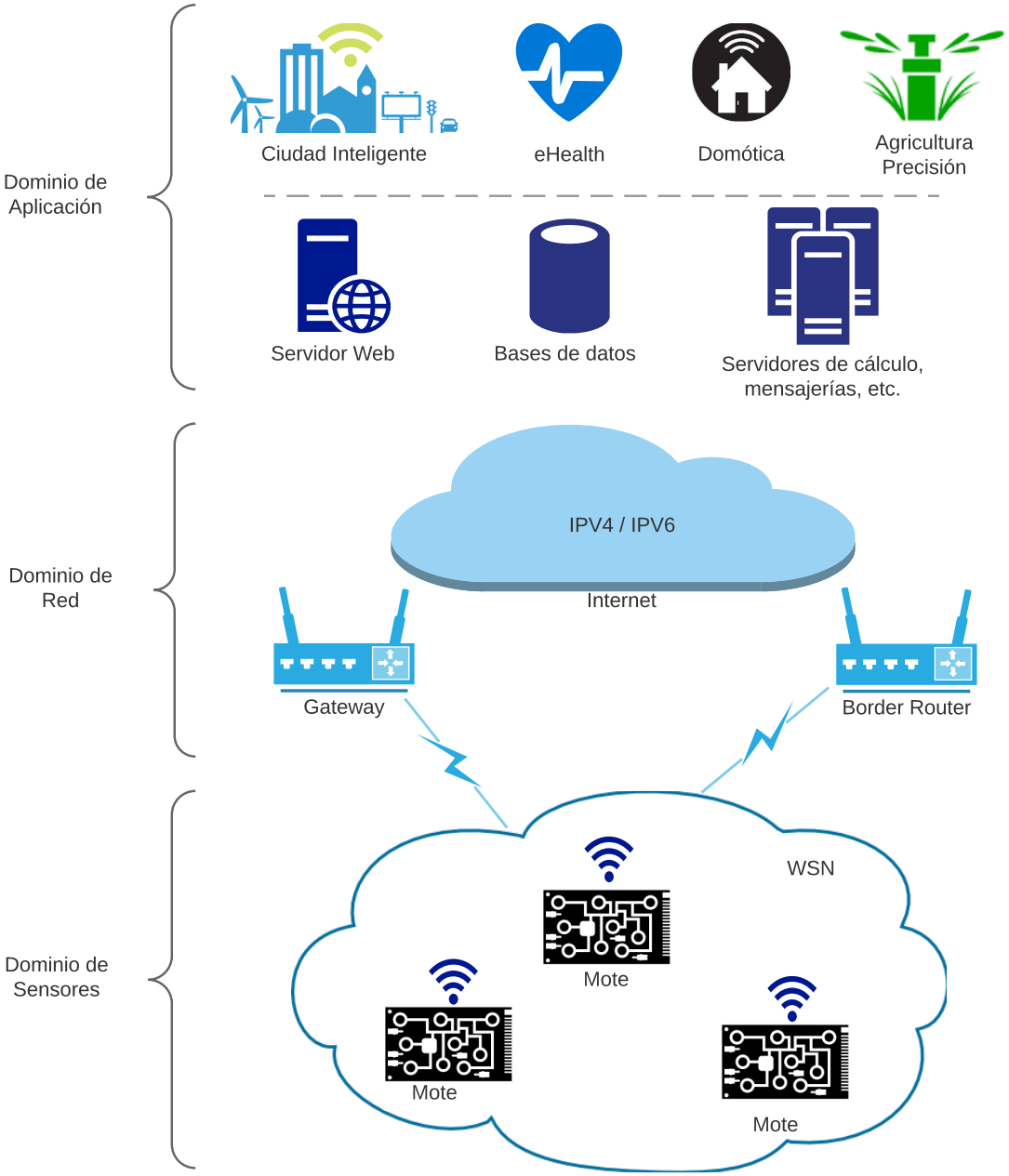


Figura 1.1: Arquitectura del Internet de las Cosas.

- Recolección de datos crudos. El CPD se comunica con la capa Dominio de Red mediante el internet y usa un protocolo de comunicación para recolectar los datos crudos [6–8]. Existen varios protocolos de comunicación, por ejemplo: Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Representational State Transfer (REST), Extensible Messaging and Presence (XMPP), etc. [9, 10]. MQTT es muy popular por su bajo consumo de ancho de banda y bajo consumo de recursos.

- Preprocesamiento y almacenamiento de datos. Consiste en la limpieza y transformación de datos para luego ser almacenados en sistemas gestores de bases de datos, y/o pasar a un sistema de cálculo o simplemente ser monitoreados y controlados en tiempo real [11–14].

- Monitoreo y control. Los datos de sensores de la Wireless Sensor Network (WSN) se presentan en un tablero de control (dashboard IoT) visual para que el usuario comprenda el estado actual de la zona o área que está vigilando. Un dashboard IoT además de monitorear sensores también puede controlar actuadores de la WSN, por ejemplo, encender o apagar una bomba, abrir o cerrar una electroválvula.

- Aplicaciones o dominios IoT. Software con interfaz web o móvil que interactúa con el usuario y a su vez con los componentes IoT. Las aplicaciones IoT también se las conoce como Smart: ciudades inteligentes (Smart Cities), hogar y edificio inteligente (Smart Home and Building), cuidado y salud inteligente (Smart Healthcare), agricultura inteligente o Agricultura de Precisión (Smart Agriculture o Precision Agriculture), etc. [15–17].

- Análisis de datos. Procesa los datos crudos obtenidos de la WSN y los combina con datos extraídos de los sistemas transaccionales para obtener información útil que ayude la toma de decisiones. La Estadística, Inteligencia de Negocios, Minería de Datos, Inteligencia Artificial, Machine Learning son algunas de las disciplinas aplicables para análisis de datos en el contexto de IoT.

El sistema que coordina y gestiona todos los componentes del dominio de aplicación de Internet de las cosas se le conoce como Plataforma IoT.

La capa de Dominio de Red comprende componentes de preprocesamiento y comunicación entre la Red de Sensores Inalámbrica (WSN) y la plataforma IoT. Los componentes IoT de esta capa son:

- Gateway o Micro data center. Es un dispositivo con características de un minicomputador que además de coordinar la comunicación con la WSN y con la plataforma IoT, se encarga de obtener los datos crudos de los dispositivos IoT o motes, luego realizar un pre-procesamiento y almacenamiento temporal y seguidamente enviarlos a la plataforma IoT mediante un protocolo de comunicación.

- Red de comunicación con la WSN. Comprende la tecnología que hace posible la comunicación entre un mote y un Gateway IoT. Ejemplos de tecnologías WSN son: Zigbee, Bluetooth Low Enegy (BLE), Long Range (Lora), Sigfox, etc., las cuales serán explicadas con más detalle en la sección de Tecnologías de Comunicación.

- Red de comunicación con la Plataforma IoT. Comprende la tecnología de comunicación del Gateway IoT con la plataforma IoT, normalmente es de Local Area Network (LAN), Wide Area Network (WAN) o Metropolitan Area Network (MAN); Estas tecnologías pueden ser: Wireless Fidelity (Wi-fi), Ethernet, Worldwide Interoperability for Microwave Access (WiMax), Long Range Wide Area Network (Lora-WAN), etc.

Algunos autores como [18–20] hablan de computación en la niebla (Fog Computing) y computación en el borde (Edge Computing), a continuación se explican estos nuevos conceptos:

- Fog Computing. Extensión de cloud computing y sus servicios al borde de la red; es decir, consiste en descentralizar de la cloud, el almacenamiento de datos, el procesamiento, los servicios y las aplicaciones para llevarlo a un entorno localizado. Esta responsabilidad por lo general es delegada a un gateway IoT de tipo MicroData Center, transmitiendo a la cloud sólo los mensajes y datos de contexto global.

- Edge Computing. La computación de este tipo va más allá de ser localizada a un Gateway o micro data center, cada dispositivo de la red desempeña la función de procesar los datos más cercanos y de decidir qué datos debe enviar al dispositivo de nivel superior (Gateway IoT o Cloud).

En la capa de Dominio de Sensores se ubican las redes de sensores inalámbricas (WSN) y los dispositivos (motes) IoT que integran transductores, sensores y actuadores.

En el argot de los sistemas embebidos y la telemetría existen tres términos que tienden a ser confundidos, estos son: transductores, sensores y actuadores dada la similitud y naturaleza que los envuelve. Según la definición de diccionarios, un transductor es un dispositivo que transfiere un tipo de energía en otra diferente. Ejemplos comunes tenemos a los micrófonos, capaces de transformar sonido en impulsos eléctricos y en el caso contrario a los altavoces o parlantes que convierten impulsos eléctricos en sonido. También tenemos a los focos de luz incandescentes que emiten luz cuando pasa corriente por un filamento. Otro ejemplo sería el motor que convierte energía eléctrica en energía mecánica o de movimiento. Por otro lado, tenemos a los sensores, que según diccionario es un dispositivo que puede detectar cambios de estímulos físicos y lo convierten en una señal que puede ser medida o guardada. Ejemplo de sensores los tenemos en el cuerpo humano, que logran detectar luz, sonidos, cambios químicos, presión y temperatura. Una acotación importante en este momento para distinguir los conceptos y términos anteriores sería que un sensor puede ser usado por sí solo para medir algo, pero un transductor necesita además del elemento de sensado un circuito eléctrico asociado.

Es decir, un transductor contiene un sensor y la mayoría de los sensores deben ser transductores. El actuador es un dispositivo que conmuta una señal eléctrica o mueve algo y utiliza energía para lograr un movimiento o conmutación. Por tanto, podemos decir que un actuador es un tipo específico de transductor.

1.1. Motivación y principales objetivos de la tesis

La necesidad de contar con un sistema de telemetría multipropósito capaz de obtener variables de un proceso en tiempo real y publicarlas en Internet, utilizando para ello hardware y firmware abierto basado en estándares, que garanticen bajo consumo, tiempo de vida alto, reducido tamaño, bajos recursos de memoria y procesador, baja latencia y que soporte protocolos de comunicación que permitan transmitir datos vía IPV6 sobre redes inalámbricas, nos motivo a llevar a cabo esta investigación.

Objetivo General

Diseñar y desarrollar sistemas de telemetría multipropósito que obtengan variables de un proceso en tiempo real y las guarde en un Cloud Computing para el Internet de las Cosas mediante una red de sensores inalámbricos formada por motes o nodos (hardware y firmware) propios o comerciales que cubran los requisitos de modularidad, alimentación, mantenimiento, escalabilidad, ultra bajo consumo, conectividad IPV6 y servicios web a nivel de nodo, basados en tecnología plug & play a nivel de sensores. Además, deberán usar estándares de comunicación que facilite interconexión con todo tipo de redes a nivel IP de forma óptima en cuanto a latencia, consumo de energía y calidad de servicio. Todo ello optimizado a través de un trabajo de investigación y análisis y desarrollo/adaptación de algoritmos para cubrir estos propósitos.

Objetivos Específicos

- Evaluar y comparar/analizar el desempeño de las tecnologías Zigbeee (IEEE 802.15.4) y Bluetooth Low Energy en el ámbito de una red WSN - 6LoWPAN para el IoT utilizando motes comerciales.

- Modificar firmware de un mote comercial seleccionado aplicando mecanismos de optimización que mejoren los resultados obtenidos.

- Diseño y desarrollo del hardware y firmware de una plataforma experimental propia que incluya las modificaciones anteriores.

1.2. Principales contribuciones de la tesis

Las principales contribuciones originales derivadas de esta tesis son las siguientes:

- Se define un conjunto de protocolos livianos que permiten la auto-detección de hardware y el registro automático en la nube con mecanismos plug-and-play que pueden implementarse en dispositivos de IoT con recursos limitados.
- Se demuestran las capacidades de los protocolos diseñados, probando el más restrictivo Lightweight Protocol for Sensors-Six (LP4S-6) después de implementarlo para una aplicación de telemetría en un beacon Eddystone, cuya especificación predeterminada se amplía para aumenta su capacidad para gestionar sensores ilimitados sin interferir en su funcionamiento normal con tramas Eddystone. Los experimentos realizados muestran la viabilidad de la solución propuesta y su superioridad respecto de los enfoques basados en el perfil genérico de atributos, General Attribute Profile (GATT), en términos de latencia y consumo de energía cuando múltiples usuarios se conectan a un sensor inteligente o en escenarios donde la latencia no es una restricción, pero el bajo consumo de energía es esencial.
- Se presenta una arquitectura basada en el estándar IEEE 21451 que propone diferentes modificaciones relacionadas con el concepto de Virtual Transducer Electronic Data Sheet (VTEDS). El sistema propuesto permite la incorporación de mecanismos plug-and-play en la capa del sensor en un ecosistema de IoT. El sistema propuesto también incluye al conjunto de protocolos livianos, antes descritos, para optimizar la descripción de los sensores mediante Transducer Electronic Data Sheet (TEDS), permitiendo la auto-detección de hardware y el auto-registro en la nube. Estos protocolos se pueden implementar en diferentes tipos de motes e incluso se pueden usar mediante protocolos estándar.
- Se presenta una aplicación web con una interfaz gráfica intuitiva que permite monitorear, controlar y administrar todos los sensores y la arquitectura de comunicaciones.
- Se evalúa empíricamente al sistema propuesto en diversos escenarios reales y con diferentes nodos de sensores para mostrar que la arquitectura diseñada es realmente rápida para desplegar e intercambiar datos de sensores.
- Se propone una arquitectura de telemetría multi-propósito para IoT flexible y escalable que permite la integración de diferentes fabricantes de sensores, actuadores, transductores inteligentes, gateways. Además de integrar diferentes tecnologías de comunicación en redes WSN y cableadas de largo y corto alcance.

- Se proponen diferentes escenarios de accesos a la cloud basados en el uso de clientes y broker MQTT.

1.3. Convenio de colaboración entre la UDC y UTMACHALA

Nuestra propuesta de investigación inicia a partir del Convenio de cooperación inter-institucional [21] subscrito entre la Universidad Técnica de Machala y la Universidad de la Coruña en el año 2014, con la finalidad de desarrollar la cooperación académica y educativa, así como promover el entendimiento mutuo entre ambas instituciones.

Dicho convenio establece el procedimiento a seguir entre la UDC y la UTMACH para la admisión y matricula de candidatos a cursar un programa doctoral. Uno de los objetivos del convenio concede a la UTMACH la facultad de seleccionar directamente a sus candidatos de conformidad con sus prioridades e intereses formativos e institucionales.

En tal sentido, el H. Consejo Universitario de la UTMACH concede su primer aval aval académico para la realización de estudios de postgrado, a 18 profesionales (actualmente hay mas de 30) que laboran en la UTMACH en calidad de docentes cuya admisión ha sido concedida por la Universidad de Coruña — España. En este primer grupo consta nuestra propuesta de investigación, avalados por la resolución No. 342/2014 [22].

1.4. Resumen de la tesis

Esta tesis está estructurada en tres partes alrededor de los sistemas telemetría multipropósito basados en el Internet de las cosas. La primera parte está cubierta por el Capítulo 2, donde se comienza con un análisis del estado del arte de los dominios de sensores, red y aplicación, el cual constituye en sí una guía rápida de selección de componentes para un sistema de telemetría actual. Seguido de una propuesta de arquitectura general de telemetría heterogénea que permita la integración de diferentes tipos de redes (WSN y cableadas), motes aislados y LAN extendida. Además se proponen diferentes escenarios de acceso a la cloud que garantizan la flexibilidad y escalabilidad deseada en los sistemas actuales IoT.

La segunda parte la cubre el Capítulo 3, donde se ilustra el desarrollo de una familia de protocolos livianos para detección heterogénea en aplicaciones de telemetría IoT y ser usados por los nodos sensores de la arquitectura propuesta en la primera parte. La familia de protocolos Lightweight Protocol for Sensors (LP4S) definida dispone de los sub-protocolos Lightweight Protocol for Sensors-Six (LP4S-6), Lightweight Protocol for Sensors-Extended (LP4S-X) y Lightweight Protocol for Sensors-JSON (LP4S-J), que están destinados a proporcionar comunicaciones livianas a dispositivos de IoT con batería y recursos limitados.

La tercera parte está cubierta por el Capítulo 4, que propone una nueva arquitectura Plug-And-Play (PnP) basada en TEDS virtuales para la Web of Things (WoT), que permite la auto-configuración, auto-descripción, auto-calibración de los nodos sensores tanto en el dominio de redes como en una cloud computing y ser usada en sistemas de telemetría multipropósito como se indicó en la primera parte.

En el Capítulo 5, se muestran las tendencias tecnológicas en el ámbito IoT que marcarán el desarrollo de trabajos futuros. También se muestran los avances de un proyecto en curso para la implementación de un sistema de telemetría basado en IoT para el control del riego automatizado de una plantación de bananos, que hace uso de los principales aportes de esta tesis.

Finalmente, en el Capítulo 6 se resumen las conclusiones de las aportaciones de la tesis y la producción científica generada por esta.

Capítulo 2

Sistema telemetría Multipropósito IoT

2.1. Introducción

Los sistemas de telemetría han sido ampliamente estudiados desde su origen en los años 1800, donde se consiguieron enviar de forma remota la medición de ciertas variables atmosféricas hacia una central de monitoreo y control. La telemetría, etimológicamente hablando proviene de las palabras griegas (tele) que significa distancia y (metron) que significa medida. Estos sistemas estaban formados por un dispositivo de entrada, que actuaba como transductor de la variable física a ser medida, unidad terminal encargada de convertir el parámetro físico medido en un dato para que un transmisor enviara dicha medición ya sea de forma cableada o inalámbrica y por diferentes medios de comunicación, un dispositivo receptor que procesaba, grababa y visualizaba los datos. Los sistemas de telemetría han encontrado un gran número de campos de aplicación con el decursar de la historia y están presentes en todas las industrias y dominios de aplicaciones actuales, como la industria del petroleo, aeronáutica, armamentista, salud, naves espaciales tripuladas o no tripuladas, automovilismo, etc., incluso la podemos encontrar hasta en los teléfonos inteligentes actuales. La telemetría ha sido utilizada no solo para obtener datos de lugares remotos sino también de acceso difícil o peligroso para el humano. Es decir, tiene una presencia constante en nuestras vidas y de una u otra forma decide e incide en el comportamiento de procesos y el progreso de la humanidad, por ende, su estudio y evolución continua sigue siendo centro de atención de la comunidad científica e industria tecnológica.

Como hemos mencionado, la industria acogió rápidamente a los sistemas de telemetría y los incorporó desde el 1960 en sus sistemas Supervisory Control and Data Acquisition (SCADA) y Distributed control Systems (DSC) para monitorear y controlar diferentes

procesos. Estos sistemas han evolucionado hasta nuestros días y con la llegada del IoT han aparecido nuevos conceptos muy en boga como son Industrial Internet of Things (IIoT), Industria 4.0 e Internet of Everything (IoE), que exigen nuevos retos a los diseñadores de sistemas de telemetría actuales, que incorporan las tradicionales funciones de los SCADA como la de adquisición de datos, comunicación, monitoreo y control, alarmas, alertas, reportes con nuevos valores añadidos gracias al IoT como son la de convertir el simple dato en información relevante incluso semántica, análisis predictivo (que es lo que va a pasar?) y prescriptivo (cuando?, por qué? y que es lo que hay que hacer?), permitiendo a su vez la creación de nuevos modelos de negocio.

En el contexto actual del IoT, los componentes básicos del sistema de telemetría antes mencionados siguen estando presentes y los encontramos de forma transversal en todos los dominios de la arquitectura IoT mostrada en la figura 1.1, esta vez en forma de sensores y actuadores, smart transducers, gateways y la cloud computing. Uno de los grandes retos de la telemetría actual está precisamente en su esencia de poder conectar cualquier cosa, problemática que se agudiza con la jungla de productores de tecnologías, protocolos de comunicación propietarios y estándares, caso que no ocurría con los diseños de SCADA hace 40 años atrás, donde prácticamente solo se conectaban componentes de un mismo fabricante.

En este capítulo, se da una solución a estas problemáticas y retos actuales dada la heterogeneidad de los elementos de un sistema de telemetría y la necesidad de convivencia en los sistemas IoT existentes. Para ello, se comienza con un estudio del estado del arte de los componentes básicos de un sistema de telemetría, basados en la publicación [23], para que sirva de guía en la implementación de un sistema IoT actual multi-propósito, incluso poniendo casos de ejemplo de un dominio específico de aplicación como es la Agricultura de Precisión. Luego, se propone una arquitectura de telemetría heterogénea, en cuanto a tecnologías y redes de comunicación (corto y largo alcance), smart transducers y métodos de acceso a la cloud. Para culminar con un ejemplo de implementación de una de las redes BLE de la arquitectura propuesta, que está basado en la publicación [24].

2.2. Estado del arte

2.2.1. Dominio de los sensores

Sensores

Los sensores típicamente convierten estímulos físicos en señales eléctricas analógicas o digitales y pueden ser clasificados de acuerdo al tipo de estímulo en: acústicos, eléctricos,

magnéticos, ópticos, térmicos y mecánicos [25]. De acuerdo a la clasificación anterior, los tipos de sensores más comunes por la variable a medir son:

- Temperatura: Termistores, Termostatos, Termocuplas y Resistance Temperature Detectors (RTD)
- Sonido: Micrófonos e Hidrófonos
- Luz: Light Dependant Resistor (LDR), Fotodiodos, Fototransistores y Celdas solares
- Fuerza/Presión: Celdas extensiométricas (Strain Gauge), Interruptores de presión y celdas de carga
- Posición: Potenciómetros, Linear Variable Differential Transformer (LDVT), Reflectivos y Encoders
- Velocidad: Tacogeneradores y acelerómetros

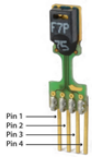
Actuadores

Los actuadores son dispositivos capaces de conseguir el movimiento de algo por medio de una energía o simplemente conmutar una corriente o un voltaje para que otro dispositivo pueda generar una acción en su entorno de un proceso dado. En función de esta energía los actuadores pueden ser clasificados en neumáticos, hidráulicos y eléctricos y en función del movimiento conseguido pueden ser lineales o rotatorios [26]. Por medio de los actuadores, un sistema automatizado puede abrir o cerrar una esclusa, activar o desactivar una electroválvula para dejar pasar agua, encender o apagar una bomba de agua, controlar el ángulo y altura de boquillas o dispensadores. Abrir escotillas de sembradoras, ajustar la cantidad de fertilizantes y dosificar el alimento de animales son algunos de los ejemplos que podríamos citar.

En la tabla 2.1, se muestran algunos de los sensores y actuadores comerciales más usados en Agricultura de Precisión, donde cada fila de la tabla responde a un tipo de sensor específico y las columnas a) y b) representan ejemplos o modelos representativos con la idea que constituya a su vez una guía inicial en la búsqueda del sensor idóneo, por ejemplo, para proyectos agropecuarios.

En la fila de sensores de temperatura tenemos en a) el sensor MCP9700A que es un sensor de temperatura, cuyo voltaje de salida es proporcional a la temperatura en un rango de -40ºC (100 mV) hasta 125ºC (1.75 V) con una sensibilidad de 10 mV/ºC. Este sensor puede ser usado para conocer la temperatura ambiental. En b) tenemos un PT -1000, que es un sensor de temperatura para el suelo principalmente, con un rango

Tabla 2.1: Sensores y actuadores típicos de Agricultura de precisión.

Sensores	Ejemplo (a)	Ejemplo (b)
Temperatura		
Humedad del suelo		
Humedad ambiental		
Radiación		
Dendrómetros		
Sensores combinados		

Actuadores	Ejemplo (a)	Ejemplo (b)
Electroválvulas		
Actuadores electromagnéticos		
Movimiento o desplazamiento		

de -50 a 300°C. El mismo constituye un sensor resistivo, donde la resistencia de salida varía entre 920 y 1200 ohms aproximadamente y a 0°C la resistencia es de 1 Kohm.

En los sensores de humedad del suelo, en a) se muestra un sensor Watermark of Irrometer, con un rango de medición de 0 a 200 cb, donde el valor de resistencia de salida es proporcional a la tensión de agua en la tierra. En b) se muestra un Tensiómetro de Irrometer, que permite medir y leer directamente la humedad del suelo. Este sensor no es afectado por la salinidad del suelo, por tanto, no necesita calibraciones adicionales.

En la fila de sensores de humedad ambiental, tenemos en a) al sensor 808H5V5 cuyo voltaje de salida es proporcional a la humedad relativa en la atmósfera y el valor se entrega en porciento de humedad relativa (0 a 100 %RH). En b) se muestra un sensor de humedad condensada, Leaf Wetness Sensor (LWS). Aquí el voltaje de salida es inversamente proporcional a la humedad condensada en el sensor. Otro parámetro importante en agricultura es la radiación solar, los sensores de este tipo se muestra en a) Light Dependant Resistor (LDR), donde la resistencia de salida es proporcional a la intensidad de luz que incide sobre la celda. En b) se encuentra el sensor SQ-110, utilizado para medir radiaciones solares y el voltaje de salida es proporcional a la intensidad de la luz, en el rango visible del espectro (410 a 655 nm).

Los dendrómetros son unos sensores interesantes que permiten medir el diámetro del tronco de una planta o del fruto para conseguir un seguimiento efectivo del cultivo. En a) tenemos al sensor resistivo Ecomatik DC2, con un rango de 0 a 20 Kohm, que mide el diámetro del tronco de una planta (Trunk Diameter Dendometer). En b) se muestra al Fruit Diameter dendometer (Ecomatik DF), presenta el mismo rango de resistencia de salida que el de tronco, pero se lo utiliza principalmente para medir el diámetro de los frutos. En la tabla 2.1 también se ha incluido una fila para transductores que combinan más de un sensor como es el caso de a) que puede ser utilizado para una estación meteorológica, incluye un pluviómetro, anemómetro y dirección del viento. El anemómetro brinda una salida digital cuya frecuencia es proporcional a la velocidad del viento. La dirección del viento (Wind vane) brinda una salida resistiva, donde la resistencia es proporcional a un ángulo, es decir brinda 16 posiciones que indicaría la dirección actual del viento. El pluviómetro, brinda una salida digital que se activa un interruptor cuando el nivel del agua ha llenado el recipient del sensor (0.28 mm) aproximadamente. En b) en cambio tenemos al sensor SHT75 de Sensirion que es un sensor digital de humedad y temperatura, con un rango de -40 °C a +123.8 °C y la humedad de 0 a 100 %RH. La salida de este sensor es una palabra digital en formato I^2C.

Las tres últimas filas de la tabla 2.1 contienen algunos actuadores usados en diversas aplicaciones, siendo una de ellas la agricultura de precisión. Siendo el control de riego una problemática de automatización en este dominio de aplicación. En esta fila en a) se muestra una típica electroválvula, utilizada principalmente para controlar el paso del agua a las zonas de riego y en b) se muestra la imagen de un interruptor de

presión para riego. Luego se ha incluido una fila de actuadores electromagnéticos que generalmente constituyen un paso intermedio de los actuadores de fuerza final, como son los contactores y relés mostrados en a) y b) respectivamente. Estos actuadores permiten a su vez accionar motores o bombas de agua, mostrado en el ejemplo a) de la última fila de la tabla y en b) se muestra el vástago lineal CAHB-20/21.

Wireless Sensor Network

Según la arquitectura IoT, analizada anteriormente en el epígrafe Arquitectura IoT, la base de esta arquitectura está formada por los transductores, sensores y actuadores que interactúan con el proceso según el dominio de aplicación que analizado en el epígrafe anterior. Estos transductores necesitan de circuitos adicionales que permitan convertir las señales analógicas medidas en digital y a su vez poder transmitirlas hacia el servidor, para que un usuario en cualquier parte del mundo pueda observar. Estos circuitos han evolucionado junto con los avances de la electrónica y las telecomunicaciones. Actualmente los transductores son conectados a dispositivos pequeños, típicamente alimentados por baterías, con ciertas limitaciones de recursos, pero con la suficiente capacidad de procesar esa señal y transmitirla o recibir comandos remotos para los actuadores. Estos dispositivos son denominados "Transductores inteligentes"(smart sensors o smart transducers) y en el argot IoT son llamados también "motes".

En la mayoría de las aplicaciones IoT, existen muchos motes interconectados en una red, es decir una red de sensores. Actualmente el medio de comunicación más usado es el inalámbrico, por lo que estas redes reciben el nombre de Redes de Sensores Inalámbricos o inglés Wireless Sensors Network (WSN). En este epígrafe vamos a introducirnos en las WSN para conocer los principales smart transducers comerciales disponibles en el mercado, que pueden ser usados en proyectos agropecuarios. También revisaremos las principales tecnologías inalámbricas usadas en las WSN actuales.

Transductor inteligente (Smart Transducer)

En la figura 2.1 se muestra el esquema general de un smart transducer. Donde podemos observar que a través de las entradas y salidas analógicas y digitales del procesador se conectan los sensores y actuadores. En un mote el procesador debe tener la capacidad de realizar un procesamiento primario de la información y encapsular el dato dentro del protocolo de comunicación que usa la WSN para comunicarse con el servidor o la cloud.

El módulo de comunicaciones muchas veces (tendencia actual) aparece integrado junto con el procesador y el bloque de acondicionamiento de la señal es opcional en dependencia del sensor y la variable a medir. Este bloque es necesario porque hay que ajustar los niveles de la señal de salida del sensor con la entrada analógica del

mote, ya que estas entradas analógicas pertenecen a un Analog Digital Converter (ADC) encargado de digitalizar la variable medida para su posterior procesamiento y transmisión. Estos ADC por lo general tienen un rango de 0 a 5VDC de entrada y las salidas de los sensores manejan rangos de mV a su salida. Por tanto está señal analógica debe ser ajustada al rango de entrada del ADC para conseguir la mayor precisión posible. El acondicionamiento de las señales analógicas se caracteriza por el uso de amplificadores operacionales con configuraciones básicas, diferenciales o incluso con amplificadores de ganancia programable inteligentes como se muestra en la figura 2.2, adaptada de [27].

La comunidad científica y la industria siempre han generado y utilizado la tecnología en aras de mejorar estándares de vida y aumentar eficiencia productiva muchas veces a través del monitoreo y automatización de múltiples procesos con diferentes grados de complejidad. Para ello ha sido siempre necesario conocerlos y caracterizarlos, para luego poder incidir sobre ellos según la lógica de decisión implementada, sistemas expertos, inteligencia artificial y machine learning, es por ende que el desarrollo de los sensores y actuadores ha sido y será una labor permanente.

Algunos de los nuevos productos de hardware están: Arduino family [28], Shield Ethernet Arduino [29], Raspberry Pi 3 [30], ThingBox [31], ESP8266 Wifi module [32], CLOUDBiT/LiTTLEBiTS [33], Particle Photon [34], Beaglebone Black [35], Pinocchio [36], UDOO Quad [37], Samsung Artik [38], nRF 52-DK [39], Espruino [40], Libelium [41],

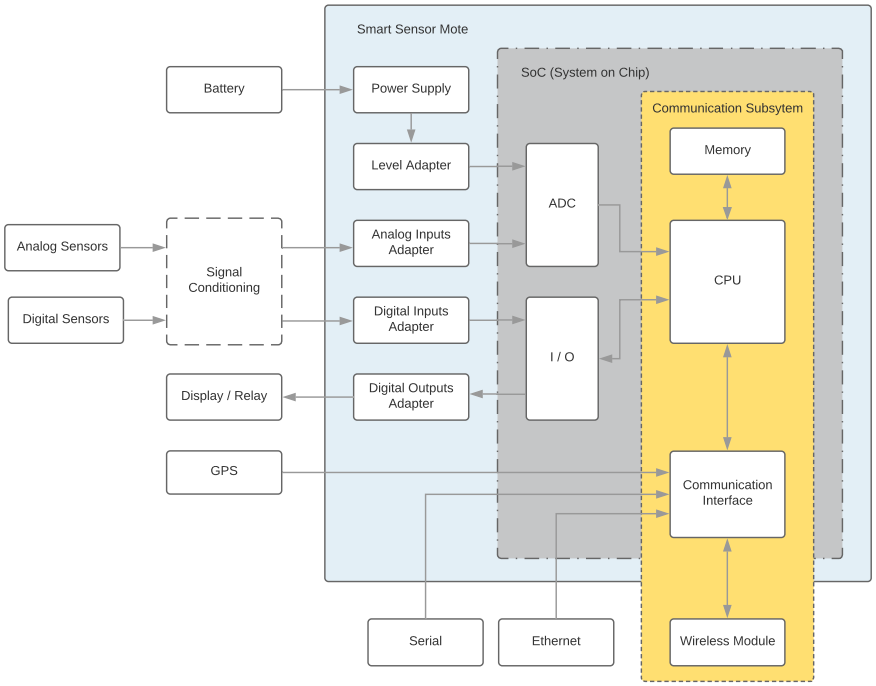


Figura 2.1: Diagrama general de un Transductor Inteligente.

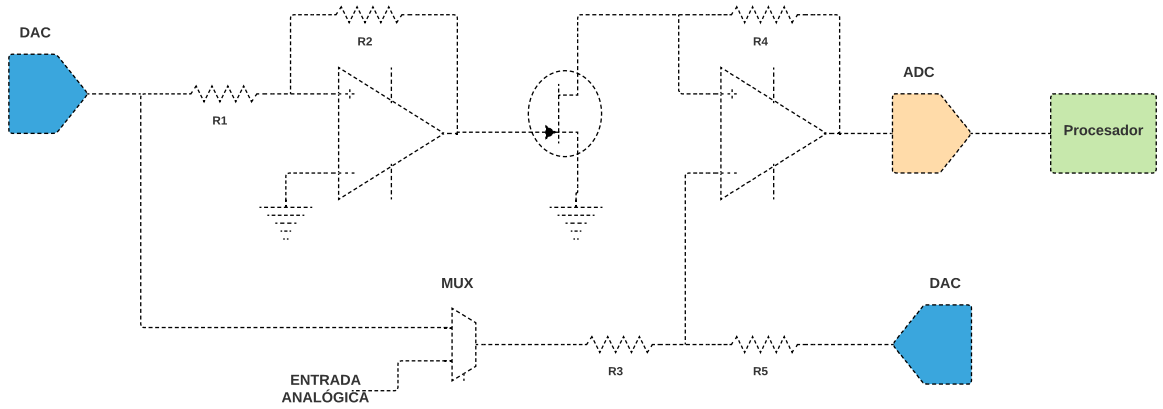
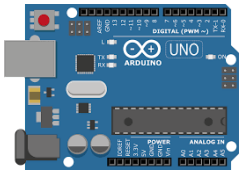
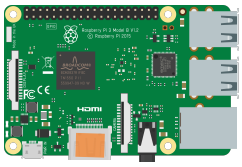
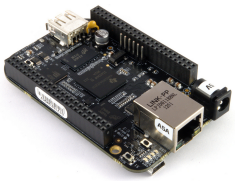


Figura 2.2: Amplificador de ganancia programable.

Microbit [42], RedBear [43] y Hexywear [44]. Por el lado de software tenemos a C y C++ como los lenguajes más usados hasta la fecha para desarrollar el firmware de los motes, aunque lenguajes como Python y JavaScript con Node-red [45] le siguen los pasos con una presencia exponencial en estas plataformas, prometiendo ser los nuevos líderes del sector. Un elemento importante en la mayoría de estos sistemas embebidos es la presencia de la arquitectura ARM, la cual ha revolucionado sin lugar a dudas el hardware de las cosas, que ha permitido embeber lightweight RTOS, como son: mbed OS [46], Android things [47], Contiki OS [48], RIOT OS [49], Zephyr [50], Apache Mynewt [51], Weave [52], Fucsia [53,54]. En [55] encontramos una excelente comparación de estos últimos.

En la tabla 2.2 se muestran algunos de los smart sensors Open Hardware disponibles comercialmente.

Tabla 2.2: Sensores inteligentes comerciales.

Smart sensor	Imagen	Características
Arduino		Microcontrolador: ATmega32u4, Voltage de trabajo: 5V, Entrada Voltage: 7-12V, Digital I/O Pines: 20, PWM Channels: 7, Canales de entradas analógicas: 12, SRAM: 2.5 KB (ATmega32u4). La configuración de conectores y señales se han convertido en un estándar. Decenas de shield o tarjetas con aplicaciones específicas han adoptado este estándar y se denominan Arduino compatible.
Raspberry Pi		Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB RAM, BCM43438 wireless LAN y Bluetooth Low Energy (BLE) embebido, 40-pines extended GPIO, 4 puertos USB 2, 4 puertos salida audio y video compuesto, Full size HDMI, Conector de cámara CSI para conectar una cámara Raspberry Pi y puerto Micro SD para cargar el sistema operativo y almacenar datos.
Beaglebone		Procesador: AM335x 1GHz ARM® Cortex-A8, 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash, Acelerador gráfico 3D, Acelerador NEON de punto flotante, Micro controladores: 2x PRU 32-bit, Conectividad con cliente USB para alimentación y comunicación, USB host, Ethernet, HDMI 2x 46 pines, compatibilidad de Software: Debian, Android, Ubuntu yCloud9 IDE on Node.js w/ BoneScript library.

Continúa en la página siguiente.

Continuación de la tabla

Smart sensor	Imagen	Características
Waspmote		Microcontrolador: ATmega1281 @ 14.74 MHz, SRAM: 8 kB, EEPROM: 4 kB, FLASH: 128 kB, SD card: 2 GB, Rango de Temperatura:[-10 ºC, +65 ºC], 7 analog inputs, 8 digital I/O, 2 UARTs, 1 I2C, 1 SPI, 1 USB, Voltaje de la batería: 3.3-4.2 V, Carga via USB: 5 V - 480 mA y Carga con panel solar: 6-12 V - 330 mA.

Tecnologías de comunicación

Las tecnologías para desarrollar IoT han evolucionado exponencialmente. Entre las tecnologías de comunicación preferidas de los desarrolladores tenemos a Long Range (Lora) [56], Long-Term Evolution (LTE) [57, 58], Sigfox [59], Wi-fi Halow [60], Bluetooth 5 [61], ANT+ [62], Thread [63], ZigBee [64], Z-Wave [65], KNX [66], Near-Field Communication (NFC) [67] y Radio-frequency identification (RFID) [68]. Entre los protocolos de Internet están HTTP (Client -server), WebSockets (Asynchronous bidirectional messaging), RESTful (COAP [69]), Pubsub (MQTT [70], Extensible Messaging and Presence (XMPP) [71], Advanced Message Queuing (AMQP) [72]) y Peer to peer (WebRTC [73]). Todas ellas han llevado al IoT a un punto sin límite de crecimiento. Como complemento a esto, encontramos las comunidades Open Hardware y Open Software con una inmensa variedad de productos disponibles y listas para el uso de los desarrolladores.

Como hemos mencionado anteriormente la arquitectura IoT está basada en redes WSN donde sensores inteligentes intercambian información entre ellos y son capaces de enviar datos de telemetría hacia el servidor gracias a los módulos de comunicación inalámbricas que poseen, muchas veces integradas con el procesador en un solo chip. Entre las tecnologías inalámbricas más usadas tenemos a: Zigbee, BLE, Lora, Sigfox las cuales son detalladas en la tabla 2.3.

Tabla 2.3: Tecnologías de comunicaciones inalámbricas.

Tecnología	Logotipo	Características
Wi-fi		Wi-Fi: Wireless Fidelity, Estándar inicial IEEE 802.11, seguidos por 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, 802.11sd,802.11ah (Wifi-Halow), Frecuencias: 2.4GHZ, 5.4GHz, 60Ghz, 900 MHz. Modos de conexión: infraestructura: dispositivo con un punto de acceso a red (Access Point), ad-hoc: red virtual entre dispositivos sin el uso de un Access Point físico, wifi-direct: conexión entre dispositivos que negocian cuál de ellos actuará como Access Point, simulando una red Wi-fi.
Zigbee		Estándar IEEE 802.15.4. Bajo consumo, Frecuencias: 868 MHz en Europa, 915 MHz en USA y 2.4 GHz en todo el mundo. Puede formar redes tipo estrella, árbol y malla. La red está formada por un coordinador, routers y dispositivos finales. Aplicaciones: Domótica, Energía Inteligente, Ciudades inteligentes, entre otros. Especificaciones: Zigbee PRO, Zigbee RF4CE y Zigbee IP

Continúa en la página siguiente.

Continuación de la tabla

Tecnología	Logotipo	Características
Bluetooth Low Energy		Estándar IEEE 802.15. Bajo consumo, Frecuencia: 2.4GHz. Utiliza Frequency Hopping para evitar interferencias con otras tecnologías que usan la misma banda de frecuencia. La última especificación es la 5.0. Velocidad de transferencia de 1 Mhz. Ha sido integrado en los teléfonos inteligentes. Redes tipo estrella. Permite crear redes de área personal Personal Area Network (PAN).
Lora		Lora: Long range (largo alcance), Plataforma inalámbrica de bajo consumo, Frecuencias: 868 MHz en Europa y 915 MHz en USA, Protocolos: Lora y LoRaWan, Velocidad baja: decenas de Kbps Grandes distancias de cobertura (algunos kilómetros).
LTE		LTE: Long Term Evolution (Evolución a largo plazo), Estándar creado por el 3GPP[11], Versión actual es Long-Term Evolution (LTE) Advanced Pro, Velocidades de transmisión de 3Gps, latencias de 2ms, Tecnología IP de extremo a extremo, Red formada por nodos eNB como estaciones bases que dispone de funcionalidad de control embebidas, evitan el uso de Radio Network Controller (RNC), Enfocada a aplicaciones IoT. Conocida como 4.5G

2.2.2. Dominio de red

Gateway y border routers

En los epígrafes anteriores se ha dado una introducción importante al dominio de los sensores, es decir los principales sensores inteligentes comerciales y tecnologías de comunicación inalámbricas predominantes en las WSN actuales. Por tanto, ya podemos subir al próximo dominio de red, encargado de garantizar que los datos provenientes de los sensores lleguen a un servidor o Cloud computing que soporte la aplicación específica de IoT.

Es necesario recordar que en Internet el protocolo aún predominante es el Internet Protocol version 4 (IPV4), con su limitante en cantidad de direcciones IP disponibles. El IoT por concepto requiere que cada dispositivo disponga de una dirección IP única, es aquí donde se requiere la implementación de Internet Protocol version 6 (IPV6) a nivel de nodo sensor, para que millones de dispositivos IoT puedan garantizar el requerimiento mencionado. Por ende, podemos intuir un pequeño problema de compatibilidad de dispositivos IoT IPV6 sobre redes IPV4 predominantes en Internet, siendo éste uno de los grandes retos encargado a este dominio. En el dominio de red, tenemos como actores principales a los Gateway y Border routers que de una u otras maneras resuelven, dan solución y soporte a este dominio. Los Border routers como su nombre lo indica son ruteadores que permiten conectar una red con otra, en este caso la WSN con Internet. Estos ruteadores se encuentran en el borde de la WSN, es decir el punto de conexión de comunicación extremo de la WSN y a ello se debe su nombre por encontrarse en el "borde" de la red.

Los gateways (pasarelas en español, aunque este es un anglicismo técnico completamente aceptado) permiten la conexión y comunicación entre dispositivos de una misma red o diferentes redes, traduciendo el protocolo de una red al nuevo protocolo que usan en la otra que deseamos conectarnos. En cambio, IoT necesita de un nuevo tipo de gateway que combine funciones del gateway y border router convencional y que brinde además seguridad, conectividad estable, traslación de protocolos, filtrado y procesamiento del dato, capacidad de almacenamiento y análisis y gestión de acceso de los motes.

Los gateway IoT actuales están basados en Windows o Linux y pueden ser implementados en diferentes plataformas de hardware, desde un teléfono inteligente, una Raspberry Pi o en plataformas propietarias más robustas como es el caso de Meshlium [74], un gateway IoT que permite conectar motes de la empresa Libelium S.L. con diferentes plataformas de cloud como se muestra en la Imagen 2.3.

2.2.3. Dominio de aplicación

Cloud Computing

Se conoce como Cloud computing o simplemente la cloud, como el acceso ubicuo a la red bajo demanda a un conjunto de recursos informáticos configurables como: redes, servidores, almacenamiento, aplicaciones y servicios [75]. También la podemos definir como el conjunto de aplicaciones y servicios que se encuentran ejecutándose en una red distribuida de recursos virtualizados con acceso utilizando protocolos de internet y estándares de red [76, 77].

En la figura 2.4 se muestra un esquema de los tipos de modelos de despliegue de una cloud:

- Nube pública. Libre acceso desde cualquier parte del mundo con posibles restricciones.
- Nube privada. Se implementa dentro de las instalaciones de una empresa (on-premise) y es de su uso exclusivo
- Nube híbrida. Combinación de una nube pública y privada al mismo tiempo.
- Nube Comunidad. Para una organización de propósito común.

Figura 2.3: Meshlium, un gateway IoT de Libelium.

Modelos de despliegue son:

- SaaS (software como servicio), infraestructura, plataforma y software de aplicación listo para su uso, en donde todo aspecto de la nube es abstracto para el usuario. Ejemplos: Google Drive, OneDrive, Dropbox, etc.
- PaaS (plataforma como servicio). Infraestructura y plataforma predefinidas a partir de las cuales el usuario puede implementar su aplicación mediante herramientas especificadas por el proveedor del servicio. Ejemplos: Microsoft Azure, Google Cloud Platform, Ecuahosting
- IaaS (infraestructura como servicio). Provee un entorno de virtualización de recursos físicos para que el usuario sea el encargado de definir una infraestructura que se ajuste a sus necesidades, los servicios IaaS más populares son: OpenStack [78] y CloudStack [79] como alternativas de cloud IaaS open source; alternativas de pago son: IBM Cloud [80], Amazon Web Services [81], Microsoft Azure [82], Google Cloud Platform [83], entre otras.

Plataforms IoT

Las plataformas IoT son sistemas computacionales de proveedores externos o desarrollados a medida, los cuales han sido creados para recibir datos de sensores, almacenarlos en sus sistemas de bases de datos y ofrecer servicios adicionales de procesamiento, análisis de datos, monitoreo de la WSN y control de actuadores. Las plataformas más destacadas actualmente son ThingSpeak [31], ThingNetworks [84], Kaa [85], Artik Cloud [86] y Temboo [87]. No debemos dejar de mencionar a las plataformas de pago, que son referentes en el desarrollo IoT, ahí encontramos a: Microsoft Azure IoT Suite [82], IBM

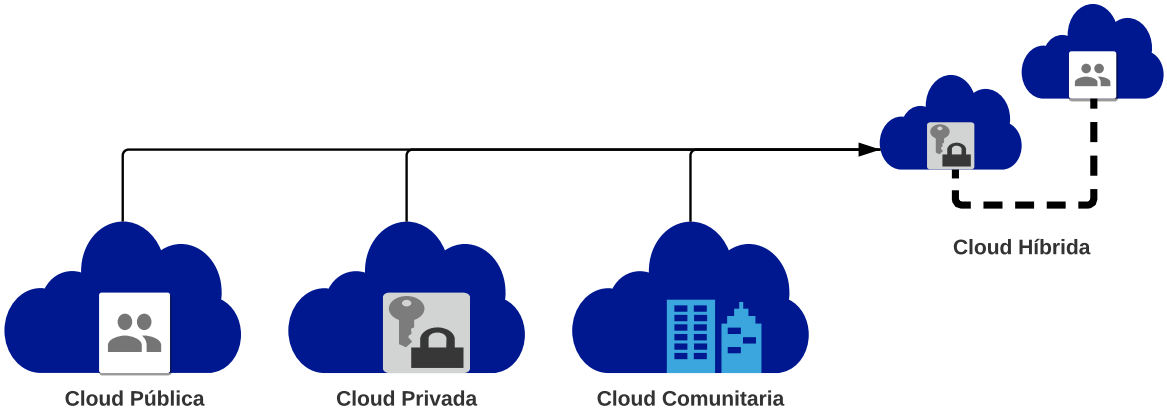


Figura 2.4: Tipos de cloud computing.

Watson IoT [88] y Xively [89]. En [90] y Postcapes [91] podrán encontrar una explicación más detallada de todos estos nuevos productos, incluyendo además una comparación de plataformas IoT que soportan los productos mencionados.

Internet of Things of UTMACHALA (IOTMACH) [92] es una plataforma IoT creada por docentes y estudiantes del grupo de investigación AutoMathTIC de la Unidad Académica de Ingeniería Civil de la Universidad Técnica de Machala, que además de monitorear y controlar redes WSN a través de dashboards creados dinámicamente, dispone de herramientas con inteligencia de negocios dedicadas a la agricultura, planeamiento y control de riego entre otras funcionalidades.

Este epígrafe ha hecho un recorrido por los aspectos más importantes del IoT que permiten al lector iniciar sus primeros pasos en la automatización ya sea de agricultura como de cualquier aplicación. Los conceptos, estructuras y plataformas indicadas en este capítulo podrán ser utilizadas en nuevos dominios IoT, a diferencia de los sensores que siempre estarán relacionados con las variables a medir del dominio en cuestión. IOT es un campo nuevo de investigación, pero con un crecimiento acelerado. Para un estudio más profundo y actualización continua se recomienda seguir la bibliografía recomendada en cada sección.

2.3. Arquitectura general de un sistema telemetría heterogéneo

En esta sección se propone una arquitectura de telemetría para IoT flexible y escalable que permite la integración de diferentes fabricantes de sensores, actuadores, smart transducers, gateways. Así como, la integración de diferentes tecnologías de comunicación, en redes WSN y cableadas de largo o corto alcance. Se proponen ademas diferentes escenarios de acceso a la cloud, para hacer factible un sistema de telemetría multipropósito heterogéneo.

En la figura 2.5 se puede apreciar la arquitectura propuesta, la cual permite ser implementada en su totalidad, forma parcial o combinaciones de redes, en función del dominio o dominios de aplicación requeridos, recursos económicos disponibles o requerimientos de diseño.

Algo importante a destacar es el uso de redes IP como común denominador de todas las redes presentes en dicha arquitectura, para hacer posible el funcionamiento integral de la misma. Por tal razón, una red LAN constituye el nodo central de toda la propuesta, con la cual se tendrá acceso al dominio de aplicación desde los diferentes dominio de sensores mostrados. Como toda aplicación IoT, se debe disponer de un enlace a Internet, a cualquier nivel en función de los recursos y requerimientos de los proyectos que deba soportar dicha arquitectura. Por tanto, gracias a Internet podrán existir sensores locales y externos. Un ejemplo clásico de esta arquitectura puede ser un Smart campus

con sensores y actuadores dentro del campus principal y otros sensores y actuadores instalados en extensiones o campus universitarios dentro de la misma ciudad o fuera de esta.

2.3.1. Tipos de Redes integradas

En la arquitectura propuesta podemos encontrar tanto redes cableadas como WSN, incluso dominio de sensores aislados con conexiones oportunistas o puntuales. Sin ser las únicas tecnologías disponibles para IoT se ha intentado plasmar las redes WSN más populares y actuales, las cuales se han clasificado en redes de corto y largo alcance.

2.3.1.1. Redes WSN de corto alcance

Entre las redes WSN de corto alcance mostradas están: Wi-fi, Zigbee y BLE. Para implementar redes Wi-fi existen varias tecnologías comerciales disponibles, desde el uso de Arduino con shield Wi-fi, módulos Wi-fi USB o smart transducer con módulos Wi-fi integrados como es el caso de los Raspberry Pi-zero W [93], cuya imagen es la que aparece como un mote-IP dentro de la red LAN, en la parte central del lado izquierdo en la arquitectura propuesta. Todos los módulos Wi-fi conectados en estrella contra el punto de acceso a la LAN del sistema, forman la red inalámbrica. En esta red, no

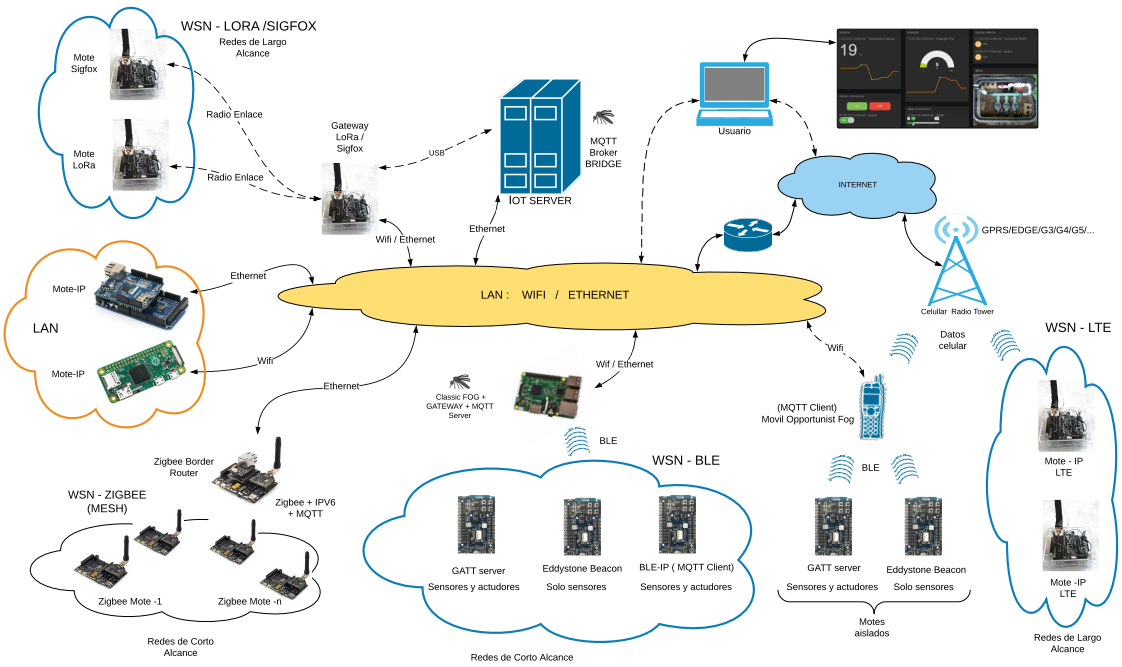


Figura 2.5: Arquitectura general de un sistema de telemetría heterogéneo.

se requiere de un gateway IoT dedicado, pero cada mote Wi-fi debe disponer de los recursos necesarios en su firmware para conectarse al servidor del dominio de aplicación.

Para el caso de Zigbee, se muestra la imagen de una solución con Waspmote Runner de Libelium [94], ubicada en el extremo inferior izquierdo de la la figura 2.5 . En este caso libelium dispone de un gateway dedicado con IPV6 over Low Power Personal Area Network (6LoWPAN), que permite crear por un lado una mesh Zigbee entre todos los motes y por otro lado puede conectarse vía ethernet a la red LAN para acceder a la cloud.

En la misma figura 2.5, se observa la imagen del kit de Nordic nRF-52DK [39], en la parte central inferior, justo debajo de la nube LAN (Wifi-Ethernet), propuesto como mote para la implementación de redes BLE, una vez que Nordic constituye el líder de fabricación e innovación de chipsets BLE. Actualmente es factible implementar 3 tipos de motes BLE, vistos de izquierda a derecha dentro de la WS-BLE, encontramos un mote que implementa un servidor GATT. Esta solución permite una comunicación bidireccional con un gateway BLE dedicado a través de Caracteristicas de Transmisión y recepción de datos. Al final de este capítulo en la sección 2.4 podrá encontrar un ejemplo de uso de este mote BLE. En el centro de la WSN-BLE encontramos una solución de telemetría novedosa de un mote BLE basado en un beacon Eddystone. En el capítulo 3 se hará un estudio extensivo de esta tecnología. En el lado derecho, encontramos un mote BLE-IP, que implementa en su firmware un cliente MQTT, habilitándolo para establecer una comunicación vía IPV6 con el dominio de red superior, formado en este caso por un gateway IoT implementado sobre una Raspberry-pi 3, encargada de crear la piconet BLE hacia el dominio de los sensores y enviar o recibir la mensajería hacia el dominio de aplicaciones a través de sus módulos Wi-fi o Ethernet integrados.

2.3.1.2. Redes WSN de largo alcance

El avance acelerado en el desarrollo de las redes de comunicación han hecho posible la aparición de nuevas tecnologías y evolución de otras en el ámbito del IoT, como son el caso de Lora/Sigfox y LTE respectivamente. Ambas han sido desarrolladas pensando en IoT, con bajo consumo de potencia, bajas latencias y pueden ser desplegadas en arquitecturas de procesadores cada vez mas restringidas, constituyendo en estos momentos fuertes candidatos a la hora de implementar sistemas de telemetría con grandes distancias entre el punto de medición de los sensores hasta la cloud. Estas redes han sido consideradas muchas veces como redes metropolitanas, pero según nuestra clasificación encajan perfectamente en redes WSN de largo alcance.

Dentro de la figura 2.5 las redes WSN-Lora/Sigfox aparecen ubicadas en el extremo superior derecho. Tanto los motes Lora como los de Sigfox requieren de un gateway para

encapsular los datos de los sensores instalados y poder alcanzar el dominio de aplicación. Los gateways disponibles actualmente permiten conexión IP por Wi-fi o Ethernet y otros solo vía USB, en este último caso el gateway debe ser conectado directamente a los servidores de comunicación de la cloud computing.

En la misma figura, pero en el extremo inferior derecho encontramos ubicada la red WSN-LTE, la cual establece comunicación con la cloud a través de las redes celulares existentes, no requieren el uso de un gateway, por tanto estos motes deben incorporar en su firmware las funciones necesarias para alcanzar la cloud directamente y a diferencia de las redes Lora/Sigfox aquí se debe pagar por el uso del tiempo-aire (airtime).

Con LTE, las compañías proveedoras del servicio de telefonía celular y transmisión IP, están recuperando el espacio que antiguamente dominaban en la telemetrías de largo alcance con las otroras redes GPRS, que revolucionaron en su momento el segmento de monitoreo y control remoto de las cosas. Al igual que los motes Lora/Sigfox, los motes basados en LTE han sido concebidos para IoT, con todas las características típicas de una WSN, pudiendo ser clasificada también como WSN de largo alcance. Para la implementación de ambas WSN, se propone el uso del último producto de la compañía Pycom, denominado Fipy [95], el cual es un poderoso mote que dispone de cinco redes integradas en el mismo hardware y completamente programables a través de MicroPython. Las redes disponibles son WiFi, Bluetooth Low Energy, Lora, Sigfox y dual LTE-M (Category M1 (CAT-M1) y NarrowBand IoT (NB-IoT)).

2.3.2. Motes asilados

En esta sección hemos ubicados aquellos motes que no forman parte de ninguna red WSN, pero tienen cada vez mayor presencia en sistemas de telemetría de casi todos los dominios de aplicación. En esta clasificación encontramos básicamente dos tipos de motes, los conectados permanentemente, por lo general con una conexión cableada vía Ethernet y aquellos que su mayor parte del tiempo pasan desconectados y ocasionalmente se conectan a un Gateway o aquellos por concepción de diseño nunca se conectan. Un ejemplo de este último son los motes tipo beacon.

En la figura 2.5, en el centro a la izquierda junto al Raspberry Pi zero W, encontramos un ejemplo de mote aislado en un Arduino con una Shield Ethernet. En la misma figura en la parte inferior derecha entre los motes BLE de corto alcance y los motes LTE, encontramos dos ejemplos clásicos de motes aislados. A la izquierda, un mote BLE con servidor GATT y a la derecha un beacon BLE Eddystone. Ambos motes pueden ser implementados con el kit nRF52-DK ya mencionados en este capítulo.

Es importante destacar que para que estos dos motes aislados se integren al sistema de telemetría propuesto, es necesario el uso de un Gateway BLE, típicamente un teléfono

inteligente móvil, que al entrar en la zona de cobertura de dichos motes pueden establecer una enlace oportunista y de esta forma los motes podrían establecer conexión con el dominio de aplicaciones. Dadas las potencialidades de los teléfonos inteligentes actuales, estos pueden disponer de funciones de gateway como de Fog computing, donde un usuario localmente podría capturar las mediciones de los sensores en tiempo real o enviar comandos a los actuadores conectados al mote con servidor GATT por ejemplo. Este tipo de fog computing fue denominado como Fog Oportunista, el cuál será explicado con mayor énfasis en el capítulo 4.

2.3.3. LAN extendida (Extended LAN)

Como hemos mencionado a lo largo de esta sección la arquitectura propuesta esta basada en el uso de redes IP, con una red LAN central que actúa como nodo central de interconexión de redes WSN de corto o largo alcance con la cloud computing. Estas redes LAN comparten el medio de comunicación, por ejemplo, Ethernet el cual permite la conexión incluso de motes aislados. Las redes LAN en sentido general son baratas y fáciles de administrar, pero tiene limitaciones físicas de implementación, Ethernet por ejemplo permite una extensión de hasta 1500 metros o limitantes en cuanto a la cantidad de hosts conectados. Una solución a este problema son los bridges, los cuales mantienen la funcionalidad de una LAN, permitiendo conectar dos o mas LANs. Al conjunto de LANs conectadas por medio de bridges se les denomina LAN extendida (Extended LAN).

En telemetría, las LAN extendidas permiten la conexión a la cloud computing tanto a WSN remotas como motes remoto aislados con capacidad IP y pueden ser implementadas usando radios enlaces de largo alcance, por ejemplo radios con tecnología Airmax de Ubiquiti Networks [96], que tienen baja latencia y ligeramente bajos costos de implementación, con excelente rendimiento tanto en enlaces Punto a Punto como Punto a Multipunto. Estos equipos son ampliamente usados por distribuidoras de Internet para establecer enlaces de ultima milla con el cliente final. Un ejemplo de esta tecnología la encontramos en la familia Nanostation [97] de Ubiquiti Networks, muy popular en enlaces punto a punto. Otra tecnología que puede ser usada como bridges de telemetría son los radios que implementan Lora / Sigfox, ya mencionados en este capítulo, con latencias superiores a Airmax, pero con consumos de potencia y costos menores. En la figura 2.6 podemos observar un ejemplo de uso de LAN extendida para conseguir la conexión a la cloud tanto a motes cableados por Ethernet como a WSN BLE remotas.

2.3.4. Escenarios de acceso a la cloud

En las secciones anteriores de este capítulo se ha expuesto una arquitectura de telemetría multipropósito que permite la interconexión de diferentes fabricantes, tecnologías, redes y protocolos de comunicación. Para conseguir esto se escogió como gestor de toda la mensajería entre los dominios de sensores y de aplicación al protocolo MQTT [70]. Este protocolo fue diseñado para trabajar eficientemente en dispositivos con severas restricciones de recursos, memoria y procesamiento de CPU. Lo que lo convirtió en la elección por excelencia para proyectos IoT. MQTT utiliza el modelo pub/sub (publicador / subscriptor), el cual permite comunicación tanto de tipo uno a varios clientes como uno a uno. Este modelo permite establecer comunicación entre todos los clientes y servidor que estén subscritos a un tópico determinado. Cualquiera que escriba en dicho tópico todos los que este escuchándolo recibirán el mensaje enviado. Esto es muy útil en telemetría, ya que permite el monitoreo de los sensores de muchos clientes conectados a la vez remotamente. MQTT garantizan la entrega de mensajería con tres diferentes niveles de Quality of Service (QoS), nivel 0, nivel 1 y nivel 2.

El servidor MQTT es denominado broker y el mas popular ha sido sin duda un broker open source denominado Mosquitto [98], el cual dispone de mecanismos internos para establecer comunicación de forma transparente al usuario entre redes IPV4 con IPV6. Lo cual es típico en IoT, disponer de motes o cosas IPV6 y acceso a la cloud a través de redes IPV4. MQTT integra también recursos de seguridad de mensajería convirtiéndolo en un sistema de tópicos robusto, con capacidad incluso de comunicación directa con un navegador de Internet y de entrega de mensajes pendientes cuando los clientes se conectan, casos típicos de motes desconectados o con sistemas de dormida profunda para el ahorro máximo de la batería de las cosas. Incluso, en [99] podemos encontrar una variante interesante de MQTT para redes WSN llamado MQTT-SN, la

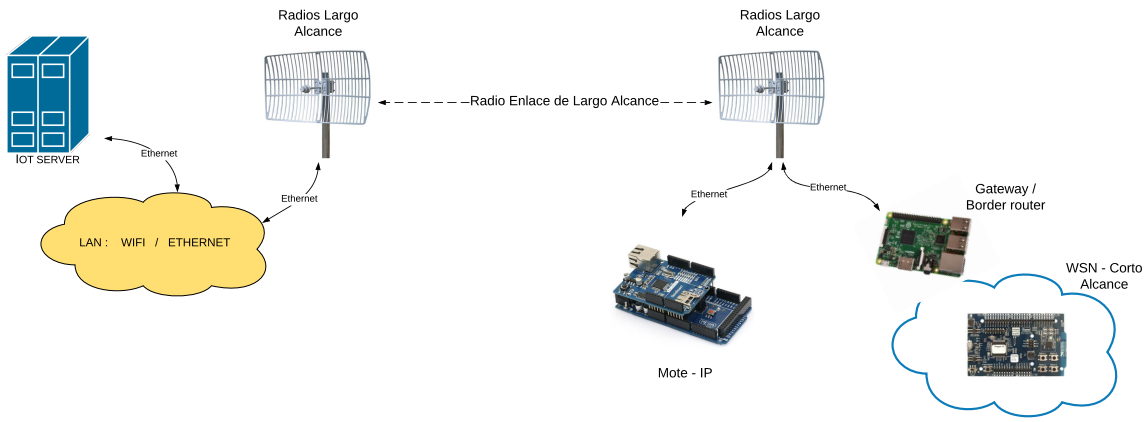


Figura 2.6: Ejemplo de una LAN extendida usando radio enlaces de largo alcance.

cual reduce considerablemente el overhead de la mensajería IP. Por todas estas ventajas nuestra arquitectura está basada en conexiones IP - MQTT, claramente reflejadas en los diferentes dominios mostrados en la figura 2.5.

Por tanto, cualquier dispositivo nuevo que se desee integrar tanto al dominio de los sensores como al de red, éste debe tener la capacidad de conexión con el servidor MQTT mas cercano disponible. La flexibilidad de diseño de nuestra arquitectura nos permite tener los siguientes escenarios de conectividad para poder acceder a los servidores IoT del dominio de aplicación:

- Conectividad directa con Capacidad IP y cliente MQTT
    1. Motes LTE
    2. Motes aislados Ethernet / Wi-fi
- Conectividad via Gateway
    1. Via Serial/USB: Lora / Sigfox
    2. Via IP - MQTT
        a) CLientes MQTT en conexión estrella
        b) Clientes MQTT aislados
        c) Mesh de servidores MQTT

La conectividad directa la consiguen todos aquellos motes, que disponen de capacidad de comunicación IP y pueden integrar en su firmware un cliente MQTT. Solo por mencionar algunos ejemplos asociados a la arquitectura propuesta, en este grupo podemos encontrar a los motes LTE y a los aislados con módulos de comunicación Ethernet o Wi-fi, que establecen una conexión tipo estrella entre los clientes MQTT de los motes con el broker MQTT del servidor, como se puede apreciar en la figura 2.7 a). Para el resto de los motes, se necesita en el dominio de red un gateway IoT, capaz de hablar en el lenguaje del mote y disponer de la capacidad IP - MQTT antes mencionada. Estos gateways pueden conectarse a los servidores de la cloud físicamente vía una conexión serial-USB, como son los casos mostrados en nuestra arquitectura para gateways Lora/Sigfox o vía IP, ya sea cableada o inalámbricamente.

En este último grupo proponemos tres formas diferentes adicionales de conectividad lógica con el servidor MQTT mas cercano. En la figura 2.7 a) encontramos el escenario de conexión estrella con el broker MQTT del servidor, aquí encontramos tanto a los motes aislados conectados directamente como a todos aquellos que dependen de un gateway móvil. En la figura 2.7 b), se muestra el escenario de una conexión estrella entre

el broker MQTT del servidor con los brokers MQTT implementados en los diferentes gateways usados, aquí se recomienda hacer una conexión Bridge disponible en el broker Mosquitto. Es importante destacar que el broker mosquito abre hilos de ejecución independientes por cada bridge activo en su configuración. A través del bridge-MQTT los motes conectados a dichos gateways pueden entablar comunicación con el dominio de aplicaciones de forma transparente. El tercer escenario mostrado en la figura 2.7 c) propone crear una mesh entre diferentes gateways con brokers MQTT conectados en modo bridge, ya no solo contra el broker del servidor sino con otros gateways. Este escenario muestra realmente la potencialidad y flexibilidad de la arquitectura propuesta, permitiendo crear diferentes niveles de fiabilidad en un sistema de telemetría ya que los mensajes podrían utilizar diferentes caminos para acceder a la cloud. Si a la vez, combinamos este escenario con LAN extendidas, las posibilidades de comunicación a nivel IoT se multiplican ostensiblemente.

2.4. Ejemplo de implementación de un testbed para IoT.

2.4.1. Introducción

Hoy en día ya no solo personas se conectan a Internet; sino también, millones de objetos o cosas (smart objects) y su número crece de forma exponencial. El Internet de las Cosas (IoT) es el internet de los objetos físicos con cierta inteligencia que les posibilita comunicarse con otros objetos y seres humanos, con el propósito de controlar situaciones críticas o relevantes para un dominio o aplicación específica. Los dispositivos (motes) integran sensores y actuadores; y, éstos están ubicados geográficamente formando redes inalámbricas (WSN) [100]. Para el monitoreo y control en tiempo real de los

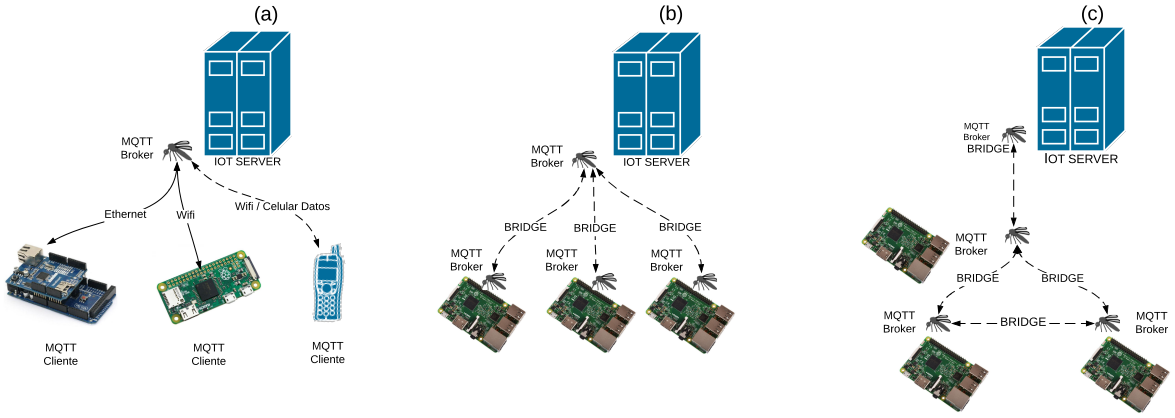


Figura 2.7: Escenarios de conexión via MQTT, (a) Estrella MQTT clients aislados, (b) Estrella con MQTT servers y (c) Mesh de MQTT servers.

smart objects, cada vez se desarrollan aplicaciones de software enfocadas a distintos dominios de IoT, algunos con mayor crecimiento son: Smart Cities, Smart Home, Smart Building, Smart Healthcare, Precision Farming or Smart Agriculture y otros [15, 101]. Las características deseadas de este tipo de aplicaciones según [102] son: automatización en la gestión, captura, comunicación, procesamiento de datos y colaboración con otros objetos [1]. El dominio IoT abordado en este trabajo es la Agricultura de Precisión (AP), conocida por algunos autores como Agricultura Inteligente y se define como el uso de las Tecnologia Informacion y las Comunicaciones (TIC) en la gestión localizada de cultivos o parcelas agrícolas [103, 104]. Un son los drones que vienen equipados con sistemas globales de posicionamiento, Global Positioning System (GPS), cámaras y sensores, que se utilizan para obtener imágenes y geodatos referenciados de una parcela o cultivo, con el fin de generar mapas de rendimiento en Sistemas de Información Geográfica, Geographic Information System (GIS). Esto ayuda al agricultor a tomar decisiones que beneficio del proceso de producción mediante la integración de los sistemas de seguimiento y control en tiempo real basado en la IoT [105]. Sin embargo, algunas de estas tecnologías están aún en desarrollo y tienen varios retos por resolver. Por ejemplo, optimizar la captura de datos y la comunicación entre los Smart objects y el Centro de Procesamiento de Datos (CPD) o cloud computing, el manejo e integración de grandes volúmenes de datos para generar información para evaluar con mayor precisión la densidad óptima de siembra, estimar fertilizantes, predecir con más exactitud la producción de los cultivos, etc. [104, 106].

Otra tecnología muy en boga es, Bluetooth Low Energy (BLE), la cual es una tecnología inalámbrica de corto alcance, también conocida como Bluetooth inteligente. A diferencia de la clásica Bluetooth presente en ordenadores, teclados y ratones; esta tecnología es de baja potencia, tamaño pequeño, de bajo costo, robusto, eficiente y permite la Interoperabilidad entre los diferentes fabricantes. Nokia creó esta tecnología y la llamó Wibree [107], para luego ser adoptado por Bluetooth SIG [108]. El desarrollo vertiginoso de BLE se debe al hecho de que los principales fabricantes de teléfonos inteligentes han incorporado a sus equipos con iOS, Android, Apple OS o sistemas operativos Windows. La comunicación entre dispositivos BLE se establece a través de una topología de red tipo estrella, donde estos dispositivos adoptan los roles de peripheral and central. Los peripheral son dispositivos pequeños con low power (motes) y el rol Central lo ocupan principalmente los smartphone, los cuales actúan como Master en la comunicación, buscando (scanning) advertisement, mientras que los peripheral actúan como esclavos haciendo el advertisment. Cuando un central device se conecta con peripheral, estos hacen uso de los GATT (Generic Attribute Profile) services and characteristics para la comunicación entre ellos. Aquí se utiliza el popular UART service con dos características, una para transmitir (Tx) y otra para recibir data (Rx).

La capa física de BLE utiliza la banda de frecuencia liberada de 2.4 GHz, compartida con otras tecnologías como Zigbee y Wifi, es por ello que se utiliza la técnica de frequency hopping para evitar o reducir interferencias.

El propósito de este epígrafe, consistió en facilitar la comunicación entre los dispositivos inalámbricos (motes) y el CPD, mediante la implementación de un Gateway para IoT basado en el sistema operativo Android que permite automatizar los siguientes procesos: 1) El descubrimiento de elementos de Redes de Sensores Inalámbricos (WSN) denominados motes (dispositivos que integran sensores y actuadores); 2) conexión con los motes mediante tecnología Bluetooth Low Energy (BLE); 3) adquisición y monitoreo de datos de sensores; 4) control de actuadores; y, 5) conexión y comunicación mediante internet con el CPD o cloud computing que almacena, procesa, controla y monitorea remotamente los dispositivos instalados en una parcela agrícola [109].

Una de las variables más utilizadas en aplicaciones de IoT es la temperatura [110–114], ya sea para medir la temperatura ambiente, suelo, agua u otro material o proceso. Los sensores Resistance Temperature Detectors (RTD) son dispositivos que varían la resistencia de un metal (principalmente Platinum) en función de la variación de temperatura, típicamente caracterizado por la ecuación Callendar-Van Dusen [115]. En este trabajo, utilizamos Platinum RTD PT1000.

2.4.2. Banco de pruebas (Testbed)

En esta sección se describe el hardware utilizado en la implementación de la WSN, la programación del mote, el proceso de desarrollo de la aplicación móvil y el protocolo de comunicación utilizado entre ambos.

Para la demostración de nuestro banco de pruebas de una WSN, elegimos el kit comercial RedbearLab, que permite hace una rápida introducción a la IoT, ya que esta tarjeta dispone de conectores pinout compatibles con Arduino (figura 2.8) y permite dos métodos para grabar el fi rmware ya sea a través de la interfaz de J-Link SWD o la plataforma m-bed. Este kit se basa en el chip de Nordic nRF51822, Nordic es el principal fabricante de chips para Bluetooth Low Energy.

Una característica importante de esta tarjeta es el modo de programación m-bed [46]. M-bed enable, significa que la tarjeta ha sido validada por el programa mbed, lo que garantiza el cumplimiento de los criterios y requisitos técnicos para la interoperabilidad con otros productos mbed enabled (comunidad Arm). Algunos de los beneficios de pertenecer a este programa son: el código desarrollado puede ser descargado en cualquier hardware con mbed enabled sin hacer ningún cambio; y la disponibilidad de una herramienta de desarrollo en línea que nos permite editar, compilar y descargar el firmware desarrollado a nuestro PC de escritorio para finalmente ser grabado en la

placa de desarrollo. Mbed tiene una gran comunidad de programadores, por lo que tiene una gran cantidad de código reutilizable. También incluye un sistema operativo que permite al programador controlar el hardware y la conexión con la nube de una manera transparente.

Con el hardware elegido, es posible evaluar el funcionamiento de gateway basado en Android. Por esta razón, se implementa un banco de pruebas que nos permite monitorear y controlar las motas WSN. Para probar los sensores analógicos, el sensor de temperatura RTD (PT1000) (DIN EN 60751) se puede utilizar dentro de un divisor de resistencia como se muestra en la figura 2.9. El voltaje proporcional a la temperatura es medido por el convertidor ADC del kit y la interpretación de las mediciones serán analizadas en la siguiente sección. Los sensores digitales se prueban con el uso de un simple botón pulsador. Para comprobar el control de actuadores, el uso de Light Emiter Diode (LED) es suficiente, ya que estos pueden ser fácilmente reemplazados por relés para controlar los motores y válvulas en un sistema de control de riego. También se añadió una salida de Pulse-Width Modulation (PWM) (Modulación por ancho del pulso) al banco de pruebas para demostrar su funcionamiento en el caso de la utilización

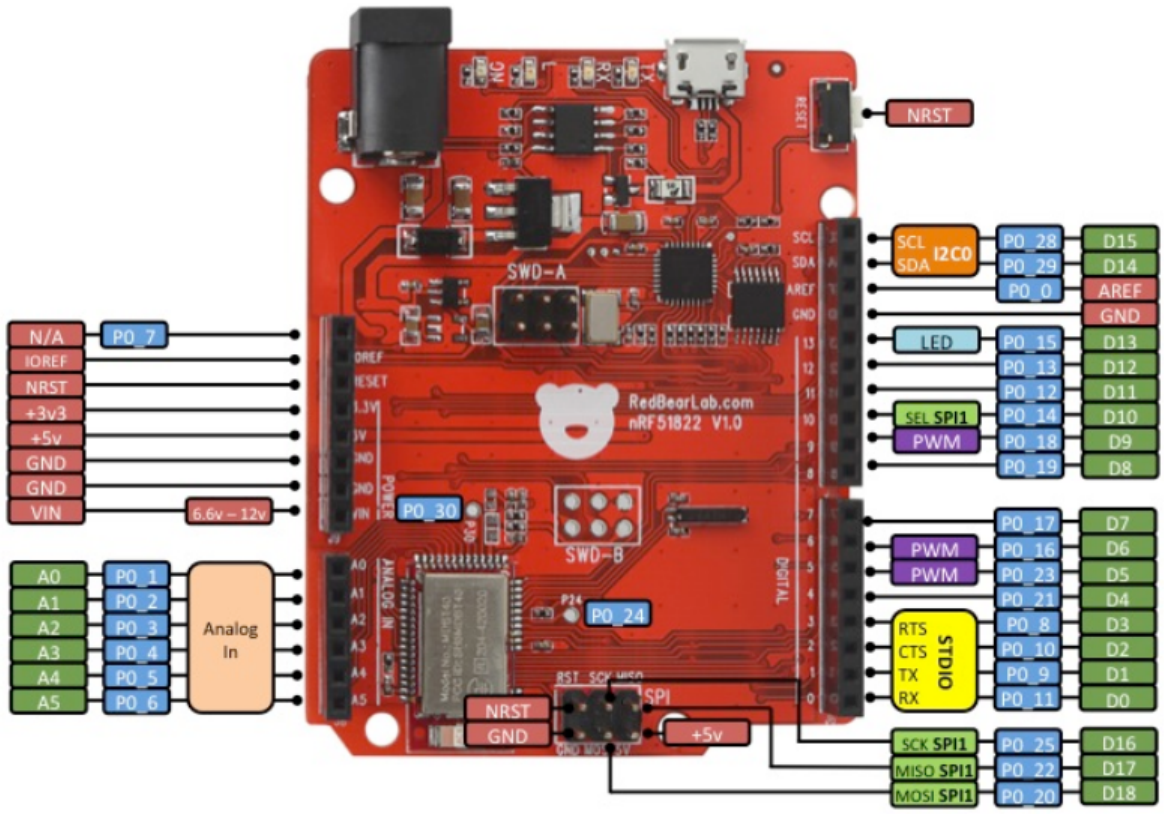


Figura 2.8: Distribución de pines del kit Redbearlab - 51822.

de motores controlables o actuadores con señales PWM. En este caso, la intensidad de luz del LED varía en función de la señal PWM enviada.

2.4.3. Conversión de Medidas en unidades de ingeniería (Pt1000)

En esta sección se analiza la lógica seguida para la interpretación de los mediciones digitales obtenidos del convertidor analógico digital (ADC) para llevarlos a unidades de ingeniría. En este caso, la variable es la temperatura y la salida se da en grados Celsius (ºC).

El procesador anteriormente mencionado utiliza un convertidor ADC de 10-bit y 3,3 VDC de voltaje de referencia (Vref) para mediciones analógicas, por lo que el valor de bit menos significativo (LSB) es igual a:

$$LSB = \frac{Vref}{2^{10\ bits}} = \frac{3{,}3\ V}{1024} = 3{,}22265625 \qquad [mV] \tag{2.1}$$

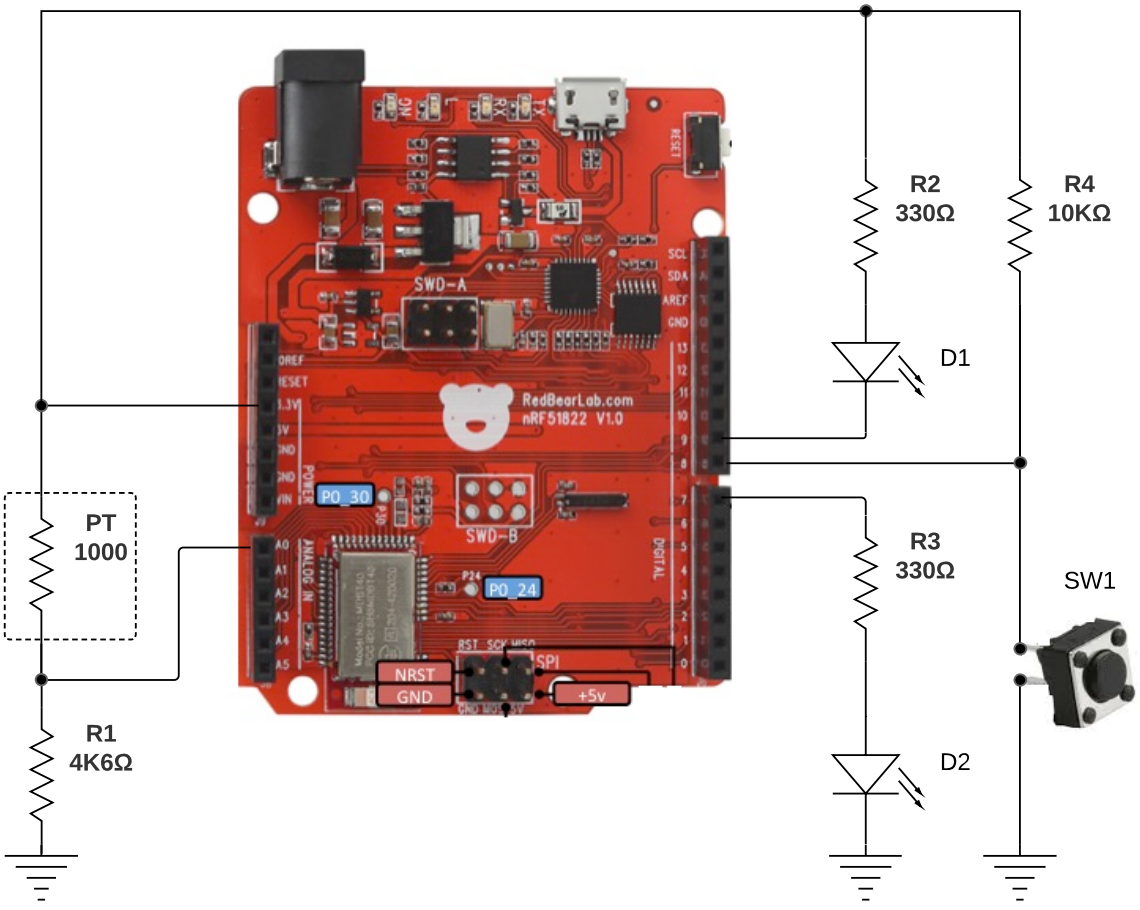


Figura 2.9: Diagrama de conexiones del testbed propuesto.

$$Vtemp = LSB \times Pd \qquad [V] \tag{2.2}$$

Por otra parte, Vtemp puede ser calculada por medio de un divisor de tensión formado por la resistencia de R4600 y el RTD (Pt1000), ver la figura 2.9. Así que:

$$Vtemp = \frac{Vref \times R4600}{R4600 + RTD} \qquad [V] \tag{2.3}$$

$$RTD = \frac{15180}{Vtemp} - 4600 \qquad [\Omega] \tag{2.4}$$

Una vez que RTD es un valor conocido, podemos determinar el valor de la temperatura usando la siguiente aproximación:

$$RTD = R_0(1 + \alpha(T_f - T_i)) \tag{2.5}$$

donde R_0 es la resistencia a 0 °C, α es el coeficiente de temperatura del PT1000 (0.00385055), T_f y T_i son las temperaturas finales e iniciales, respectivamente. Con la siguiente expresión obtenemos el valor de la temperatura en unidades de ingeniería:

$$T_f = \frac{RTD - 1000}{3{,}85055} \qquad [^oC] \tag{2.6}$$

2.4.4. Programación Mote

Dado que el BLE utiliza el servicio Universal Asynchronous Receiver-Transmitter (UART), la programación del mote se basó en el BLE UART service. La aplicación del mote fue programada en el lenguaje "C" y consta de la clase principal main, donde se inicializan y configuran los principales parámetros de BLE y la aplicación en sí. El resto de las funciones del mote serán atenciones a los eventos que van a ocurrir como son: polling de muestreo de los sensores y envió de datos, desconexión del mote con algún dispositivo bluetooth low energy y ejecución de los comandos enviados desde el dispositivo BLE pareado con el mote, como podemos ver en la figura 2.10.

Función principal

En esta función se configuran los parámetros necesarios para el correcto funcionamiento del testbed, como son: tiempo de muestreo para las mediciones, creación de "handlers" y "callbacks" para cada evento de la aplicación. Luego de esto el resto de las operaciones del mote serán por eventos.

Función de Desconexión

Esta función atiende el evento de desconexión, cuando el dispositivo que estaba pareado con el mote se desconecta por cualquier motivo y básicamente lo que hace es reiniciar el proceso de "Advertising". En la figura 2.11 a) se muestra el diagrama de la lógica de programación de esta función.

Función Muestreo

Esta función atiende el evento del Tick del sistema operativo, por tanto cada vez que ocurra un "tick" se ejecutara esta función, que será usada para leer las entradas analógicas y digitales correspondientes, como se aprecia en la figura 2.11 b).

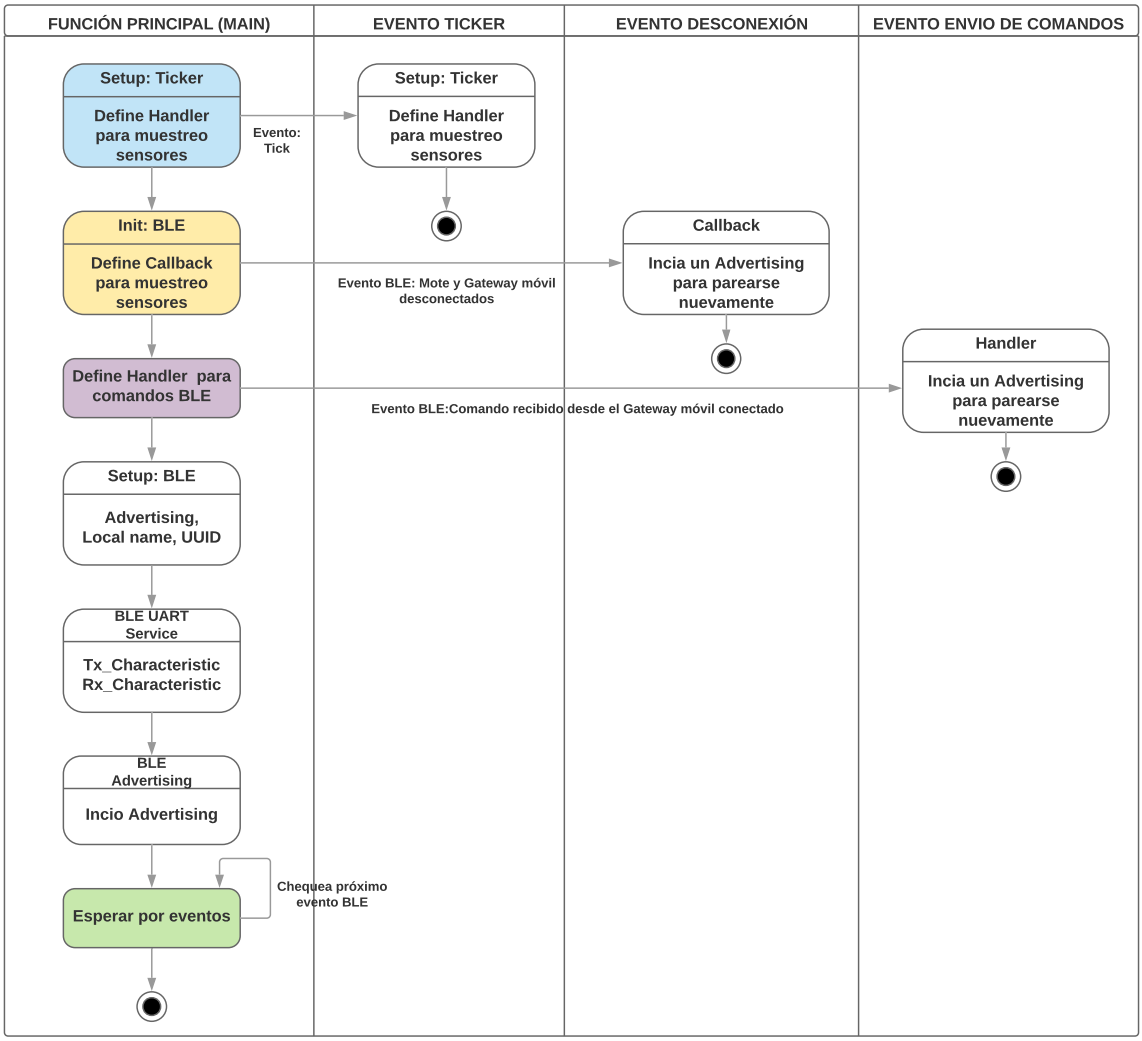


Figura 2.10: Diagrama de actividades del firmware del mote.

Función Comandos

Esta función atiende el evento cuando el dispositivo pareado envía datos o comandos vía BLE, primeramente, se extrae los comandos y datos de la "tx-characteristic" según el protocolo de comunicación establecido. Luego se ejecuta la operación modificando la variable de salida del mote correspondiente, ya sea una salida digital encendiendo o apagando un LED, o una salida PWM (Modulación por ancho del pulso) con el valor definido en la interfaz de usuario del celular. Ver figura 2.11 c).

2.4.5. Implementación de la Aplicación para Android

En la figura 2.12 se muestra el flujo de trabajo del sistema de telemetría desarrollado, el cual tiene tres componentes principales: mote, gateway y servidor. El mote o "thing" encargado de interactuar con el proceso específico, que en la aplicación actúa como peripheral haciendo advertising cada 100 ms, a la espera de un central device para parearse. Una vez que una central device lo descubra y se conecte a él, ambos, central and peripheral pasan a un estado llamado paired. Una vez aquí, la comunicación se realiza a través del profile generico UART, con una characteristic (Tx) para enviar datos desde el mote al gateway, y otra characteristic (Rx) para recibir los comandos.

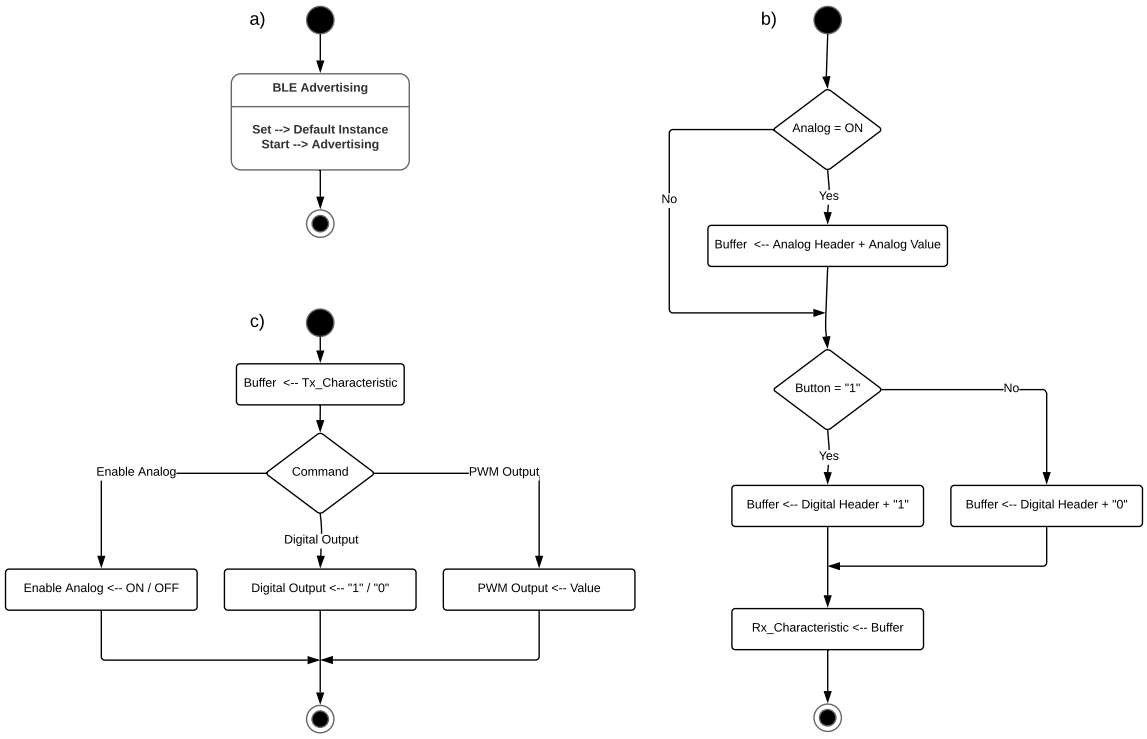


Figura 2.11: Diagrama de flujo de los eventos del mote.

La aplicación en Android (App) tiene doble funcionalidad: como monitoreo y control de la WSN y como gateway. Para ello se verifica que el servicio de BLE del smartphone esté activo, caso contrario se termina la App. Luego, se dispone de varias pantallas o interfaces para la autenticación en el servidor, interactuar en tiempo real con los motes de una WSN y graficar en tiempo real datos de sensores. Todas estas funciones accesibles a través de botones funcionales. Luego se muestra en un listado, todos los motes descubiertos para que el usuario seleccione con quien desea conectarse. Una vez que el smartphone entre en estado de paired, la App permite interactuar directamente con el mote, y como gateway, estableciendo una pasarela de datos desde el mote hacia el server y viceversa a través de un mecanismo de sincronización de ambos extremos con la persistencia de datos y comandos en la base de datos local. Por último, tenemos al servidor, donde un usuario final a través de un dashboard desde un browser tradicional puede monitorear remotamente las variables de un proceso en tiempo real, así como enviar remotamente comandos para abrir o cerrar una electroválvula, encender un LED, etc. Esto es posible en el sistema gracias a los mecanismos de sincronización implementados en el servidor.

2.4.6. Resultados y discusión

Siguiendo el diagrama de flujo descrito en la sección anterior, un sistema de telemetría fue implementado, desarrollando firmware para un BLE mote, un gateway IoT y una cloud computing. Los resultados de esta implementación serán mostrados en está sección.

Funcionalidad de la aplicación Android

La aplicación Android fue implementada con dos requerimientos: monitoreo y control in situ de una WSN y para actuar como Gateway entre los motes BLE y la cloud computing, que permite a los usuarios acceder a los datos adquiridos del dominio de aplicación IoT (Agricultura) desde la web y enviar comandos remotos, por ejemplo encender o apagar una bomba de agua o una electroválvula.

La figura 2.13 muestra varias capturas de pantalla de la aplicación Android que demuestran el cumplimiento de los objetivos planteados. En la figura 2.13 a) se muestra la pantalla el menú principal de la aplicación, para acceder a las funciones principales del gateway, dígase la gestión de dispositivos descubiertos, configuración del gateway y de los parámetros de comunicación necesarios para establecer conexión con un servidor remoto. En la figura 2.13 b) se muestra la interfaz que permite habilitar los servicios disponibles en el gateway, en la imagen está seleccionado el servicio GATT server para habilitar las caracteírticas de Tx y Rx mencionadas en secciones anteriores para establecer el intercambio de información entre el mote seleccionado y el gateway. En la

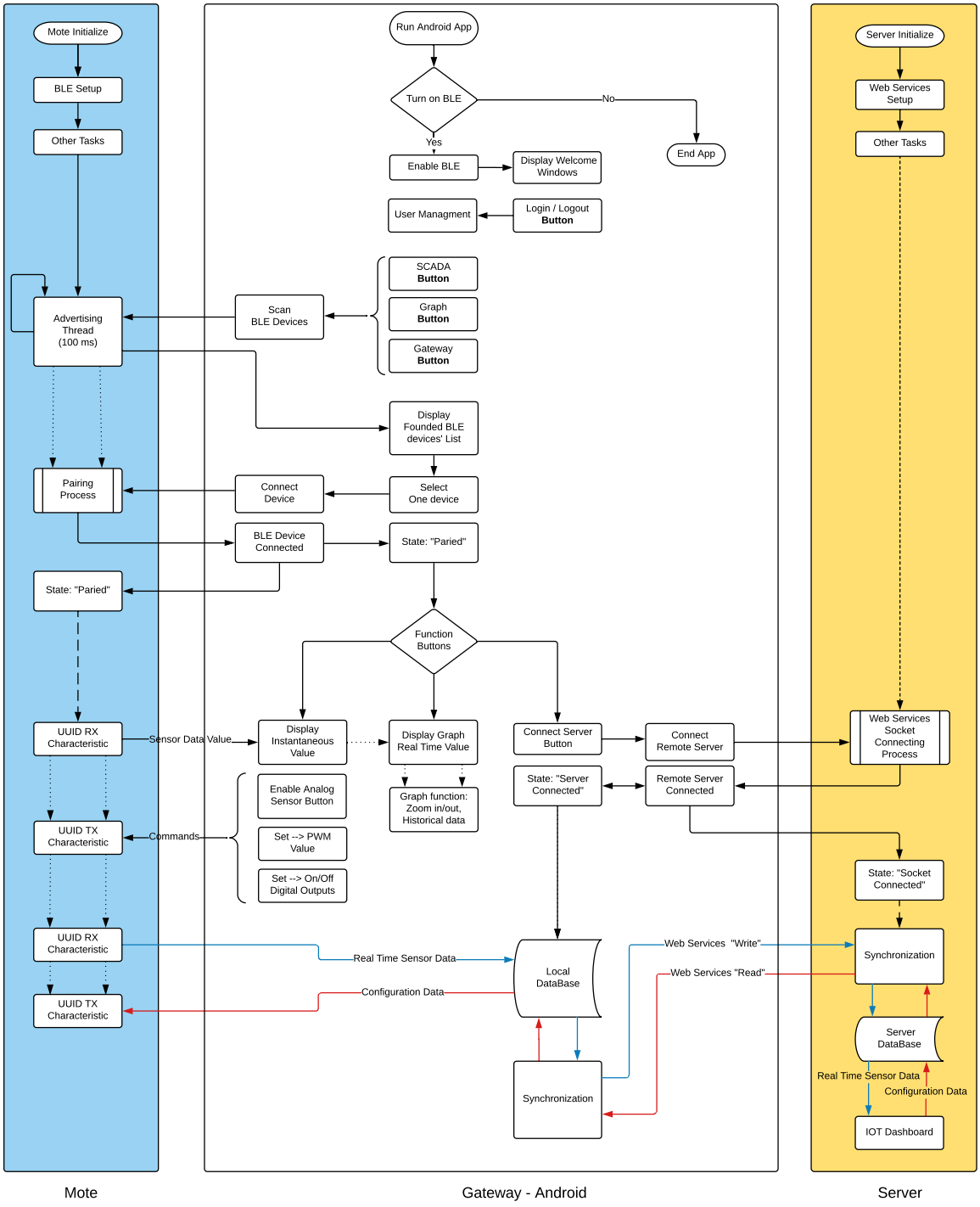


Figura 2.12: Diagrama de flujo del sistema de telemetría del testbed.

figura 2.13 c) se muestra la interfaz que permite configurar los parámetros necesarios para hacer una conexión con un servidor remoto de la cloud. En d) se muestra la primera pantalla donde aparecen todos los motes de la WSN descubiertos. Al dar click

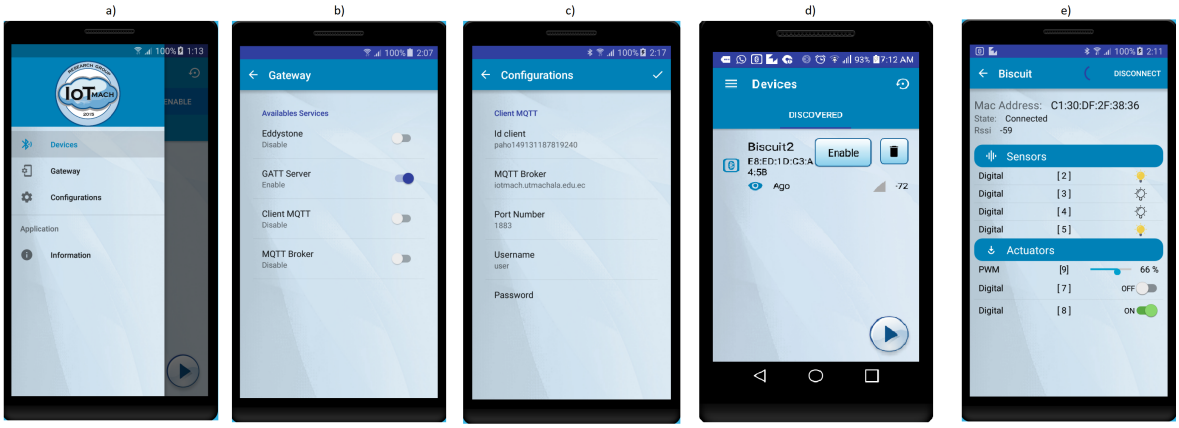


Figura 2.13: Capturas de pantalla del Gateway móvil Android para IoT.

sobre uno de ellos se establece una conexión entre el mote y el smartphone que permite visualizar valores instantáneos de las variables que está midiendo el mote, así como los controles necesarios para enviar comandos a los actuadores, como se aprecia en la figura 2.13 e).

Experimento 1: Tiempos de respuesta

Al igual que en la demostración de la funcionalidad de la aplicación, los experimentos aquí abordados están de acuerdo con los requisitos de diseño. Los experimentos realizados se llevan a cabo para medir la tasa de respuesta del banco de pruebas propuesto para ambas soluciones local y una solución completa IoT, por ejemplo, el tiempo de respuesta desde el sensor hasta la nube. El rendimiento del Gateway también se mide en términos de la utilización de la Central Processing Unit (CPU) y la memoria del teléfono inteligente con diferentes cargas de mensajes.

El escenario de prueba propuesto consta de tres componentes: Mote, aplicaciones móviles y servidor IOTMACH. Se utilizaron las siguientes herramientas para la recopilación de las mediciones: Software TeraTerm para capturar el momento en que el mote envía paquetes por el puerto serial con cada nueva lectura del sensor. El TeraTerm se ejecuta en el PC donde el mote (RedbearLab) está conectado a través de un puerto serie USB. El programa Android 2.3.3 Estudio también fue utilizado en el PC para tomar los registros enviados por el teléfono inteligente (Samsung J500 M, la versión de Android: 6.0.1) conectado a través de USB al PC. Para capturar el tiempo de llegada de los mensajes en el servidor provenientes del gateway, se utilizó el software MQTT SPY, conectado al mismo broker MQTT del servidor, donde el Gateway también se conecta para enviar sus mensajes. La conexión entre el Gateway y el mote es a través

de Bluetooth Low Energy y la conexión con la nube (IoTMACH Sever) es a través de Wi-Fi. Una visión general de este escenario se muestra en la figura 2.14.

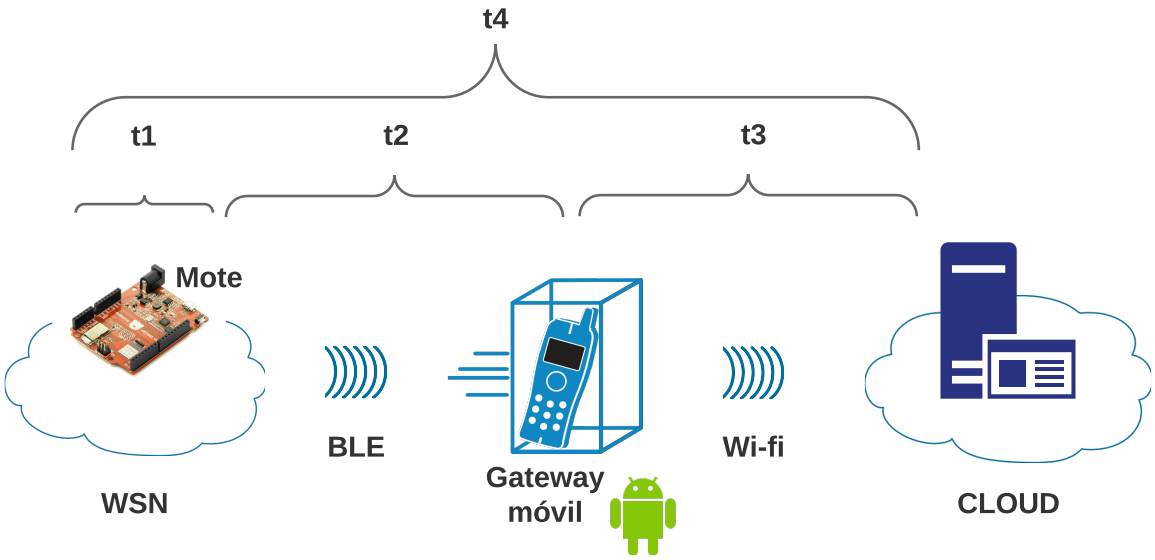


Figura 2.14: Escenario del testbed IoT.

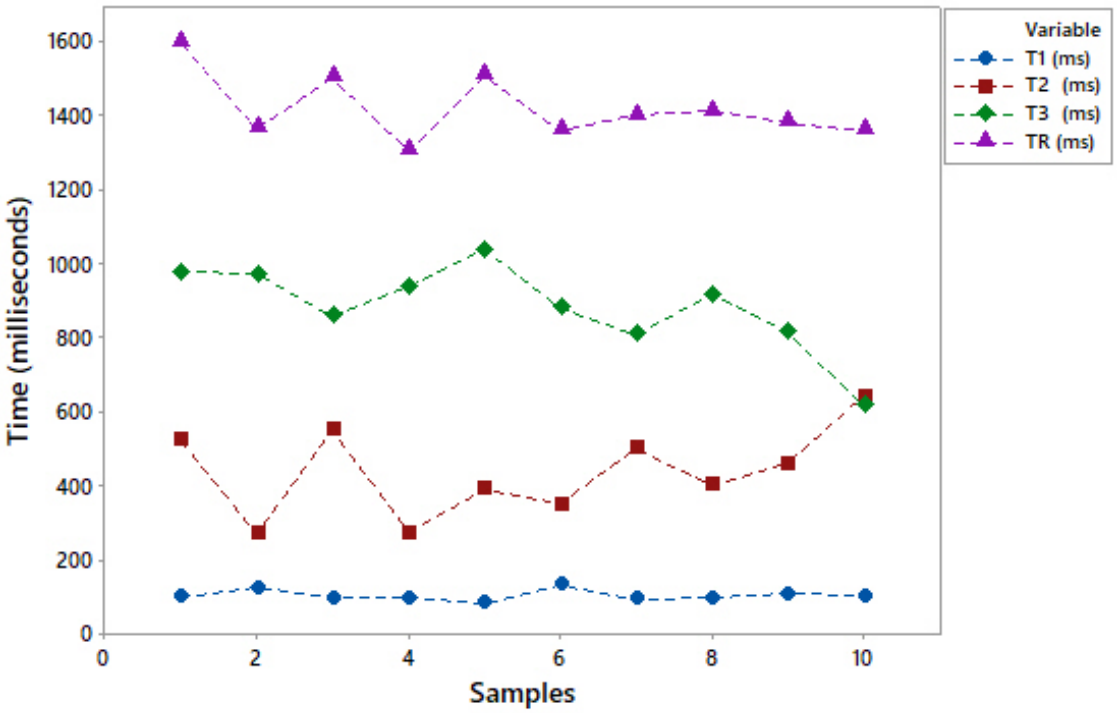


Figura 2.15: Tiempos de respuestas por conexiones y total.

Tabla 2.4: Tiempos de respuesta del experimento 1.

Variable	Promedio	Desviación estandard	Mínimo	Q1	Q2	Máximo
t1 (ms)	101.5	14.93	80	93.5	111	130
t2 (ms)	435.1	121.3	269	329.8	529	640
t3 (ms)	880.9	118.3	626	812.3	972	1036
t4 (ms)	1417.5	88.8	1304	1357.8	1503	1594

Tabla 2.5: Rendimiento del teléfono inteligente actuando como gateway IoT.

Cantidad mensajes	Promedio uso CPU (%)	Desviación estandard (uso CPU)	Promedio uso RAM (Mbytes)	Desviación estandard (uso RAM)
10	11.79	2.161	10.80	0.102
100	13.27	1.000	13.94	1.104
1000	13.30	1.097	23.87	1.414

Los resultados de los experimentos enviando decenas de mensajes desde el mote hacia el servidor se pueden observar en la figura 2.15, donde t1 representa la latencia que introduce el mote para empaquetar los datos del sensor. T2 representa el promedio de la latencia del medio de comunicación (BLE) y t3 es la demora para el intercambio de datos entre el Gateway y el servidor. Estas latencias pueden variar en dependencia del medio usado, ya sea Wifi, GPRS, 4G o y en dependencia obviamente del plan de datos del smart phone usado. T4 representa el tiempo total desde la lectura del sensor hasta que este valor llegue a la cloud. En la tabla 2.4 podemos encontrar los detalles estadísticos de los experimentos, en los que podemos resaltar la velocidad de nuestro testbed, con un tiempo de respuesta promedio de solo 1.4 segundos. Este tiempo es excelente para aplicaciones de Agricultura de precisión donde las variables como temperatura, humedad, etc., varían lentamente en el tiempo.

Experimento 2: Rendimiento del gateway

En el segundo experimento, se midió el rendimiento del gateway. La aplicación Android se modificó para aumentar la carga de mensajes entre el Gateway y el servidor, lo que permitiría un total de 30 observaciones separadas en tres series con 10 iteraciones para cada uno. En la primera serie, se enviaron 10 mensajes, en la segunda serie, 100 mensajes y en la tercera serie, 1000 mensajes. El uso de las funciones internas del software Android Studio, permitieron obtener el uso del CPU y memoria del smartphone para cada una de las series mencionadas. Los resultados se muestran en la figura 2.16 y detalles estadísticos en la tabla 2.5 , donde podemos destacar que el uso de la CPU es prácticamente constante, independientemente del número de mensajes gestionados, con una pequeña variación promedio entre 11,79 % y 13,30 %. El uso de la RAM muestra

un comportamiento lógico proporcional a la cantidad de mensajes transmitidos y que tienen que ser procesados en el Gateway.

Finalmente podemos concluir, que la aplicación Android desarrollada abordada en este epígrafe, descubre las motes BLE que se encuentra en su entorno y cuando establece conexiones con dichos motes. La aplicación permite la vigilancia y el control in situ de los sensores y actuadores conectados al mote. Además, actúa como gateway al permite la recepción de los mensajes procedentes de la WSN y encapsularlos para ser transmitidos a través de MQTT a la nube. Esto permite que los usuarios de diferentes dominios de aplicación, como es el caso de la agricultura de precisión, puedan de forma remota y a través del Internet gestionar sus procesos mediante el control de variables tales como la temperatura y la humedad; o mediante la manipulación de electroválvulas y bombas de agua para controlar la irrigación de algún cultivo. Los resultados de nuestros experimentos demuestran que se dispone de un gateway rápido, con baja latencia y excelente comportamiento en smartphones comerciales. El bajo uso del CPU en el gateway permite que el teléfono inteligente quede libre para realizar otras funciones y operaciones sin afectar el rendimiento del gateway. El costo proporcional de la memoria RAM ofrece la posibilidad de manejar altas tasas de mensajería o varias motas. Siguiendo el flujo de trabajo del sistema de telemetría mostrado en este epígrafe,

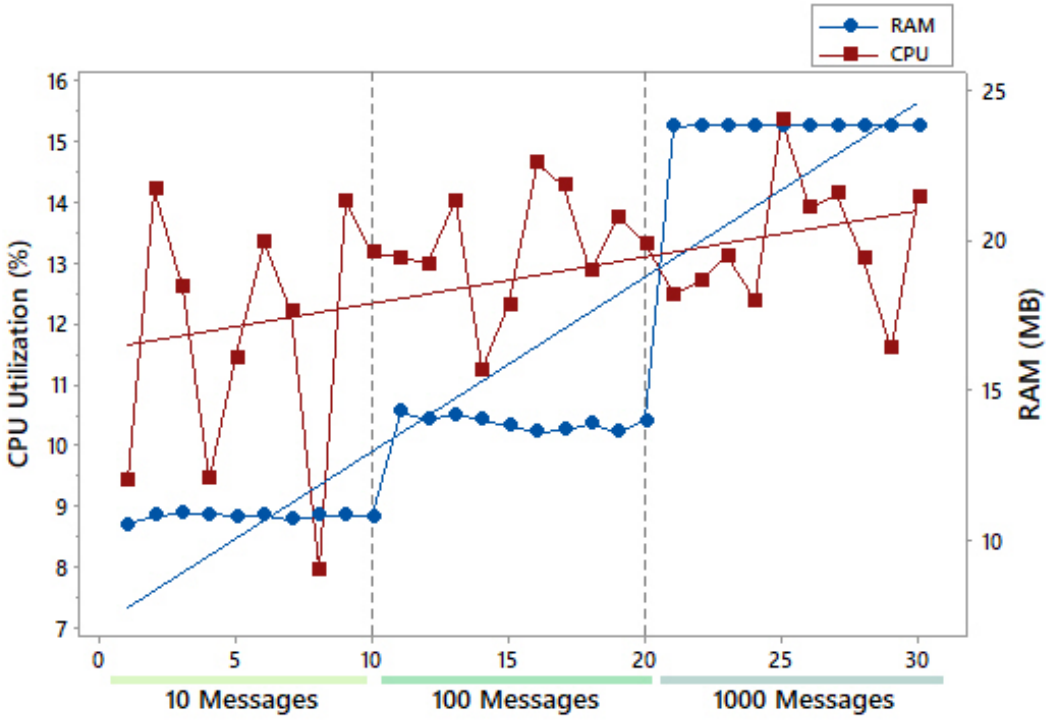


Figura 2.16: Rendimiento del gateway en función del uso de la CPU y RAM.

este banco de pruebas puede ser implementado en otros sistemas operativos de teléfonos inteligentes.

2.4.6.1. Conclusiones

En este capítulo se propuso un sistema de telemetría multi-propósito IoT teniendo en cuenta la heterogeneidad actual de las tecnologías involucradas en los dominios de sensores, red y aplicaciones. Antes de ello, se analiza el concepto de telemetría, su presencia en sistemas SCADA y DSC, así como la evolución de estos hasta llegar a conceptos actuales de Industria 4.0, IIoT e IoE. Luego se revisó el estado del arte de cada dominio, comenzando por sensores, actuadores, transductores inteligentes y tecnologías de comunicación, mostrando ejemplos comerciales representativos de cada uno, para que sirva como guía o punto de partida en la implementación de sistemas de telemetría IoT actuales. El estado del arte continuó con el gateway y diferentes modelos de despliegue de cloud computing. Aquí también se ofrece una guía rápida de elección de plataformas IoT actuales disponibles en el mercado.

Dentro de la arquitectura general propuesta se propone la interconexión de diferentes tipos de redes, corto y largo alcance, motes aislados y LAN extendida por medio del protocolo MQTT. Diferentes escenarios de acceso a la cloud son analizados, exponiendo la flexibilidad y escalabilidad de dicha arquitectura. Por último se muestra un ejemplo de implementación de un testbed IoT, desarrollando firmware para motes BLE y un gateway basado en teléfono inteligente con sistema operativo Android.

En los próximos capítulos serán expuestos dos elementos novedosos que hacen posible la implementación real de esta arquitectura, el primero con una familia de protocolos ligeros diseñados para IoT para la comunicación entre todos los elementos y el segundo con una descripción novedosa del dominio de los sensores partiendo de estándares ya definidos.

Capítulo 3

Familia de protocolos livianos para detección heterogénea en aplicaciones de telemetría IoT

3.1. Introducción

La telemetría ha sido uno de los impulsores del desarrollo de diferentes tecnologías destinadas a mejorar la eficiencia de las aplicaciones tradicionales de automatización. Por ejemplo, el Internet de las cosas (IoT) también depende de la existencia de servicios de telemetría fiables. Por otra parte, la importancia de la IoT se ha incrementado notablemente en la última década, con un máximo de 15,4 millones de dispositivos instalados en la IoT 2015 que se espera llegar a 30,7 billones en 2020 y 75,4 billones en 2025 [116].

La mayoría de los sistemas actuales de IoT están basados en redes de sensores inteligentes que están principalmente conectados de forma inalámbrica, conformando lo que se conoce como una red de sensores inalámbricos Wireless Sensor Network (WSN) [1]. Ciertos nodos de tales redes se llaman motas o motes, debido a su pequeño tamaño y la potencia de cálculo limitada. Las motas pueden ser conectados directamente a Internet o a través de gateways o enrutadores de borde (border routers), pudiendo integrar servidores web que permiten la creación de una web de sensores Web of Things (WoT) [117].

Después de considerar las tecnologías inalámbricas disponibles, este capítulo se centra en Bluetooth Low Enegy (BLE), que se ha convertido recientemente en la elección de muchos desarrolladores de la IoT por las siguientes razones principales:

- Se ha incorporado masivamente en los teléfonos inteligentes.

- Se apoya en los sistemas operativos principales (por ejemplo, IOS, Android, Linux, OSX y Windows).
- Es una tecnología estándar que es capaz de correr sobre IPv6 [118].
- Sus características han mejorado progresivamente. Por ejemplo, la última especificación, Bluetooth 5 [119], expande el rango de cobertura de 50 a 200 m y su velocidad se ha duplicado con respecto a Bluetooth 4.2.

En los últimos años, solo en dispositivos wearables, los beacons han tenido el ritmo de crecimiento más rápido dentro del ecosistema BLE. Los beacons basados en BLE son dispositivos que transmiten paquetes de datos físicos periódicamente con el fin de indicar su presencia o para transmitir cierta información. Ellos son comúnmente utilizados en servicios basados en la localización. Algunos servicios [120] han informado de que en el último trimestre de 2016, 8 de los 13 millones de sensores registrados en su red, eran beacons [121]. ABI Research en su último pronóstico [122] predice envíos anuales de dispositivos Bluetooth para llegar a 5 mil millones en 2021. Aunque los smartphones todavía representarán el 43 % de los envíos de dispositivos Bluetooth en este momento, Bluetooth Smart, también conocido como BLE, está exhibiendo el crecimiento más fuerte con un Compound Annual Growth Rate (CAGR) predictivamente de un 34 % entre 2016 y 2021, impulsado por las nuevas oportunidades de los beacons, domótica y aplicaciones weareables en las que el bajo consumo de energía es crítico. Como resultado, los dispositivos Bluetooth Smart representarán el 27 % de los envíos totales de Bluetooth para el año 2021.

Aunque los beacons tienen muchas ventajas (es decir, tamaño pequeño, bajo peso, bajo costo y bajo consumo de energía), estos están restringidos en términos de memoria, batería y recursos de hardware. Debido a estas razones, los beacons por lo general requieren el uso de protocolos ligeros en las diferentes capas del stack IoT que les permitan extender la vida útil de la batería, disminuir la latencia y garantizar la calidad de servicio (QoS). Estos protocolos tienen que minimizar las comunicaciones, la carga de cálculo y las necesidades de almacenamiento con el fin de ser implementado en beacons. Por otra parte, el uso de protocolos estándar proporciona escalabilidad, interoperabilidad y funcionalidad a los sistemas IoT actuales. Por lo tanto, este tipo de protocolos ligeros tienen que ser:

- Compatible con los protocolos ya existentes.
- Flexible para ser implementado en beacons, weareables o gateways más robustos.
- Escalable para permitir su uso en aplicaciones de nuevos campos emergentes.

Las necesidades actuales de la IoT y WoT requieren estos protocolos para ser capaz de utilizar mecanismos plug-and-play para descubrir, describir y registrar automáticamente los sensores y actuadores conectados a dispositivos inteligentes IoT. Además, es importante tener en cuenta que los beacons son dispositivos pequeños, no robustos y de bajo costo de hardware que permiten la transmisión de datos y notificar su presencia a los dispositivos Bluetooth cercanos. Por lo tanto, son capaces de transmitir mensajes e información pertinente a los dispositivos móviles que se encuentren dentro de un rango específico. Los beacons fueron concebidos para diferentes aplicaciones como telemetría, por lo que tienen restricciones con respecto a la cantidad de sensores y actuadores que pueden ser gestionados sin alterar la forma en que operan.

Con el fin de solucionar las problemáticas mencionadas anteriormente, este capítulo incluye las siguientes contribuciones principales:

- Se definen un set de protocolos ligeros (llamados Lightweight Protocol for Sensors (LP4S)) que permiten auto-detección de hardware y auto-registro en la cloud con mecanismos plug-and-play que pueden ser implementados en motes con bajos recursos y capacidades de hardware.

- Se presenta y evalúa un sensor inteligente de telemetría basado en un beacon tipo Eddystone [123], beacon BLE abierto lanzado por Google en julio de 2015. La solución hace uso del protocolo LP4S más restrictivo, lo que permite la creación de aplicaciones de telemetría con un gran número de sensores y actuadores sin afectar el funcionamiento regular del protocolo de Eddystone.

Este capítulo está basado en [124] y está organizado de la siguiente manera. Sección 3.2 revisa el trabajo relacionado con los protocolos livianos y los beacons BLE. Sección 3.3 describe los elementos principales de la familia de protocolos LP4S. Sección 3.4 detalla la implementación del protocolo LP4S-6 en un beacon Eddystone. Sección 3.5 describe y analiza los resultados de varios experimentos que evalúan el rendimiento del protocolo LP4S-6. Finalmente, Sección 3.6 está dedicado a las conclusiones.

3.2. Trabajos relacionados

En esta sección se revisan los trabajos relacionados de la comunidad científica que nos permitan determinar los requerimientos básicos esperados para dispositivos IoT y cuales han sido los protocolos ligeros definidos para dispositivos IoT.

3.2.1. Requerimientos básicos para dispositivos IoT

El IoT ha permitido un monitoreo y supervisión mas efectivo, mejorar la toma de decisiones y la eficacia en sectores como la salud (eHealth) [125, 126], ciudades inteligentes [127], defensa [128,129] y aplicaciones de alta seguridad [130–132], agricultura de precisión [133], transporte [134–136] o Industria 4.0 [137–139].

Los sistemas IoT actuales se caracterizan por el uso de dispositivos electrónicos con recursos limitados, memoria reducida, capacidad de comunicación limitada, potencia de cálculo baja, y una elevada dependencia de las baterías. Por lo tanto, no pueden apoyar la aplicación de los regímenes de seguridad complejos [140]. Estas restricciones se aplican también a los beacons y weareables, cuyo diseño representa un desafío para la comunidad de desarrolladores la IoT [141, 142].

De hecho, algunos autores [143] describen a los dispositivos IoT ideales como elementos de cálculo auto suficientes (self-sufficiente) y de eficiencia energética (energy-efficient) que proporcionan un rendimiento ininterrumpido, calidad de servicio y la duración de la batería. Para crear este tipo de dispositivos ideales, las soluciones ideadas tienen que centrarse en tres áreas principales: la eficiencia energética [144–146], la optimización de la programación y protocolos ligeros. Por lo tanto, nuevos protocolos ligeros deben ser diseñados para reducir el costo de comunicaciones (es decir, para reducir al mínimo el número de mensajes intercambiados), coste computacional (es decir, la realización de operaciones ligeras) y el costo de almacenamiento [147].

El trabajo presentado en este capítulo se centra en el desarrollo de nuevos protocolos ligeros que cumplan con los requisitos antes mencionados y que implementen mecanismos plug-and-play, que serán detallados en las siguientes secciones.

3.2.2. Protocolos ligeros para dispositivos IoT

Los primeros protocolos ligeros se dirigían a reducir la sobrecarga en la transmisión de datos en las capas inferiores de la pila de comunicaciones. Sin embargo, como se propone en [148], no es suficiente para hacer uso de ellos en las capas física y Medium Access Control (MAC). También se necesitan protocolos de peso ligero en la capa de aplicación, lo que permite reducir el consumo de energía mientras se transmiten los mensajes necesarios. Por lo tanto, en los últimos años, un gran progreso se ha logrado y podemos encontrar protocolos ligeros para prácticamente cada capa de la pila de comunicaciones.

En cuanto a la capa MAC, Chen presenta en [149] una propuesta interesante para el desarrollo de un protocolo WSN MAC ligero para una casa inteligente (smart home). Dicho protocolo reemplaza IEEE 802.15.4, que es complejo y tiene una baja relación de paquetes de entrega (low packet-delivery ratio). Los autores probaron el protocolo

mediante el uso de una matriz Field-Programable Gate (FPGA) y fueron capaces de aumentar la relación de paquetes de entrega hasta el 100 %. Otros investigadores [57] han Contribuido en varias capas de la pila con implementaciones para Contiki [48] al usar un protocolo de aplicación restringida (Constrained Application Protocol (CoAP) [150] y 6LoWPAN (IPv6 sobre redes de área personal inalámbricas) [151].

Con respecto a la capa de red, una de las primeras revisiones de los protocolos ligeros para IoT se presenta en [152]. En el documento, los autores hacen hincapié en el papel de 6LoWPAN para llevar a cabo una implementación que incluye el uso de los servicios Representational State Transfer (REST) y nodos basados en el sistema operativo Tiny. Otra solución es presentada en [153], donde se combinan los protocolos de red y la capa de seguridad, integrando DTLS [154] Con 6LoWPAN en el llamado 6LoWPAN-NHC, que codifica diferentes encabezados DTLS. Los investigadores implementaron CoAP sobre Contiki, a pesar de que afirman que se puede implementar en cualquier otro sistema operativo que soporte 6LoWPAN.

El desarrollo de técnicas seguras de autenticación ligeras sigue siendo un reto para la comunidad IoT, ya que deberá lograr un equilibrio entre un alto nivel de seguridad, eficiencia, coste computacional y costo de comunicación. Algunos autores ya han abordado algunos de estos retos [155]. Por ejemplo, en [156] Se presenta un esquema de autenticación robusto basado en COAP para el control de los recursos en un entorno IoT. Otro ejemplo interesante se puede encontrar en la referencia [157], donde se propone el uso de un protocolo de criptografía ligero basado en operaciones XOR. Además, vale la pena mencionar que algunas de las últimas revisiones de los protocolos de autenticación ligeros para weareables se pueden encontrar en la referencia [147, 158, 159].

Entre todas las capas, la capa de aplicación es probablemente la que ha tenido recientemente un mayor impacto en el crecimiento de las IoT gracias al desarrollo de varios protocolos ligeros exitosos como MQTT, CoAP, XMPP y AMQP [143, 148, 160–162].

Varios autores han propuesto protocolos para las capas de aplicación y presentación. Por ejemplo, con el fin de obtener un protocolo ligero para la WoT, Cheah implementa en [163] el Web Thing Model (WTM), propuesto por el W3C [164] en prototipos basados en COAP para descubrir y describir los diferentes elementos de un entorno de hogar inteligente. Otra propuesta interesante es el protocolo ligero Lightweight M2M (LWM2M), que es apoyado por Open Mobile Alliance (OMA) y se describe en [165] (Un ejemplo de aplicación se puede encontrar en [166]).

La contribución de los trabajos mencionados en las diferentes capas de la pila IoT se resume en la Tabla 3.1. Aunque muchas de estas obras tienen enfoques transversales, se han agrupado en capas específicas. Para una mejor comprensión se incluyen en algunos de los protocolos predominantes en cada capa sin ser estos el objeto de estudio de

Tabla 3.1: Trabajos relacionados en las diferentes capas de la pila IoT.

Capas	Protocolos	Referencias
Applicación	IoT, WoT, SWoT, Device Management, Analytics, Business Process	[162, 165, 166]
Presentación	Binary, LWM2M, XML, JSON, IPSO Objects, SOAP	[162, 163, 165, 166]
Sesión	CoAP, MQTT, XMPP, AMQP, HTTP, Websocket	[143, 148, 160–162]
Transporte	UDP, TCP, IPSP, GATT, GAP, DTLS, TLS	[147, 157–159]
Red	IPV6/IPV4, RPL, IPSP, 6LoWPAN Adaptation	[152, 153]
Física/Enlace	IEEE 802.15.4, IEEE 802.3, Ethernet, Wi-Fi, BLE, Z-Wave, DASH7, RFID	[57, 149]

nuestro trabajo. Nuestro protocolo puede ser ubicado entre las capas de aplicación y presentación.

3.2.3. Fundamentos de comunicación Bluetooth Low Energy

En los últimos años, se han propuesto diferentes tecnologías para proporcionar conectividad inalámbrica a WSN para habilitar entornos de telemetría basado en sensores IoT como Bluetooth [167], Zigbee [64], Wi-Fi, Low Frecuency (LF)/High Fecuency (HF)/Ultra High Fecuency (UHF) RFID [168–170], O incluso soluciones multi-tecnología [171].

Entre ellos, Bluetooth es una de las tecnologías inalámbricas con el mayor crecimiento y penetración en las industrias de la electrónica, de telecomunicaciones y de consumo. Su presencia en los periféricos, tales como teclados, mouse, audífonos y teléfonos inteligentes han garantizado su uso en el corto y mediano plazo. Tres versiones de Bluetooth se pueden distinguir: BLE, Bluetooth Basic Rate/Enhanced Data Rate (Bit Rate (BR)/Enhanced Data Rate (EDR)) and Dual Mode (BR/EDR/BLE). Un dispositivo que implementa este último puede comunicarse tanto con BR/EDR y dispositivos BLE. BR/EDR se conoce como *Classic Bluetooth*, mientras que el modo dual se llama *Smart Ready* y BLE es llamado simplemente *Smart*. Cada implementación se ha diseñado para usos específicos y diferentes chipsets [172].

Las diferentes implementaciones siguen las especificaciones emitidas por el Grupo de Interés Especial Bluetooth (SIG), que ha publicado versiones 2.0 y 2.1 para BR/EDR y 4.0, 4.1, 4.2 y 5.0 para BLE.

Se puede afirmar que BLE es la versión IoT de Bluetooth, ya que su bajo consumo de energía es perfecta para los dispositivos dependientes de batería, ya sea alimentados por baterías tipo botón AA/AAA o por algún tipo de solución de energía harvesting.

Los sistemas BLE trabajan en la banda no licenciada de 2,4 GHz y utilizan saltos de frecuencia (frequency hopping) para minimizar la interferencia de otros dispositivos de RF que operan en la misma banda (por ejemplo, Wi-Fi o Bluetooth clásico). Con el fin de comunicar dispositivos BLE entre sí, uno de ellos tiene que tomar el papel de

Maestro y el otro, el papel de esclavo (Máster - slave). El Máster envía paquetes tipo advertisiment en intervalos específicos de tiempo a través de los canales 37, 38 y 39 de la banda de 2,4 GHz, para que un esclavo puede descubrirlo y establecer una comunicación entre ambos. La especificación v5.0 del estándar [119] define estos paquetes periódicos como eventos que suceden después de un cierto número de unidades de tiempo. Cuatro tipos de eventos se definen: Advertising, Extended Advertising, Periodic Advertising and Connection events.

Hay dos tipos básicos de comunicaciones de dispositivos BLE en IoT: la primera uno se realiza a través de GATT, con sus servicios y características, mientras que la segunda está relacionada con beacons. El primer método de comunicación requiere establecer conexión con el dispositivo BLE descubierto, mientras que los beacons son sin conexión (es decir, que sólo tienen que ser descubiertos). Los protocolos propuestos en este capítulo están diseñados para trabajar tanto en las comunicaciones del GATT como en beacons, tal como se explicará más adelante en la sección 3.3. Una descripción más profunda de la norma BLE está más allá del alcance de este documento, pero los lectores interesados tienen un excelente punto de partida en [172].

3.2.4. Beacons BLE

Los beacons BLE son pequeños dispositivos alimentados por baterías que son capaces de transmitir cierta información (por ejemplo, la posición, url, estado de la batería, los valores de los sensores conectados) sin establecer una conexión con un escáner BLE. Esta información se envía en los paquetes advertising de acuerdo con las especificaciones SIG. Por ejemplo, un teléfono inteligente que actúa como un escáner puede ser capaz de obtener de un beacon, su ubicación, y puede generar notificaciones (es decir, alertas), recibir mensajes o activar los servicios cuando se acerca a él [173]. Un escenario típico con beacons se ilustra en la figura 3.1, donde los mensajes de difusión de un beacon son leídos por varios escáneres (es decir, receptores).

Aunque Bluetooth SIG no ha definido un estándar oficial para beacons, hay tres implementaciones principales: iIBeacon [174], Altbeacon [175] y Eddystone, que han sido definida por Apple, Radio Red, y Google, respectivamente. La figura 3.2 muestra la estructura interna de los paquetes enviados por tales protocolos.

La familia iBeacon fueron los primeros beacons en entrar al mercado. Estos fueron diseñados bajo la filosofía de código cerrado de Apple, concebidos para ser utilizados solamente con iPhones, iPads y iPods a través de sus programas de certificación iBeacon. Sin embargo, muchas personas comenzaron a utilizar la tecnología iBeacon en los dispositivos Android a pesar de estas restricciones. Esta tendencia llevó a Radius Network a crear un beacon de código abierto llamado Altbeacon, que difiere de iBeacons

en su estructura de paquetes, pero que es compatible con ellos y proporciona una funcionalidad similar. Dos años después del lanzamiento de los primeros beacons, Google entró en el mercado con su beacon Eddystone, respaldado por su amplia comunidad Android. Google, a diferencia de Apple, proporciona soporte para Eddystone para el sistema operativo iOS. Es importante señalar que, como se muestra en la figura 3.2, el protocolo de Eddystone hace uso de cuatro marcos (frames):

- Unique Identifier (UID): marcos que emiten un código único que, cuando se asocia con cierta información en una aplicación, puede ser útil para identificar lugares u objetos.
- Ephemeral Identifier (EID): marcos efímeros, son una variante de los marcos de UID que ofuscar el ID para mayor privacidad y seguridad. Este marco no se ha

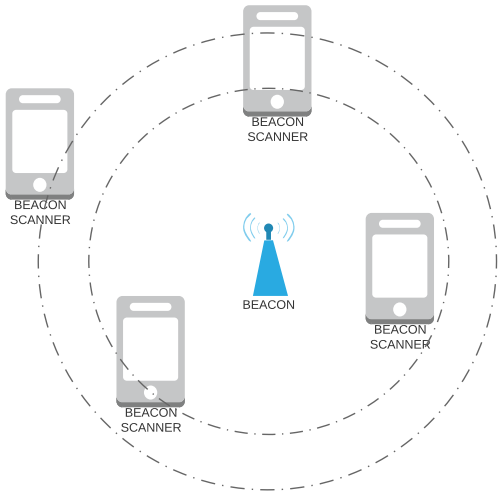


Figura 3.1: Escenario con múltiples escáneres de beacon.

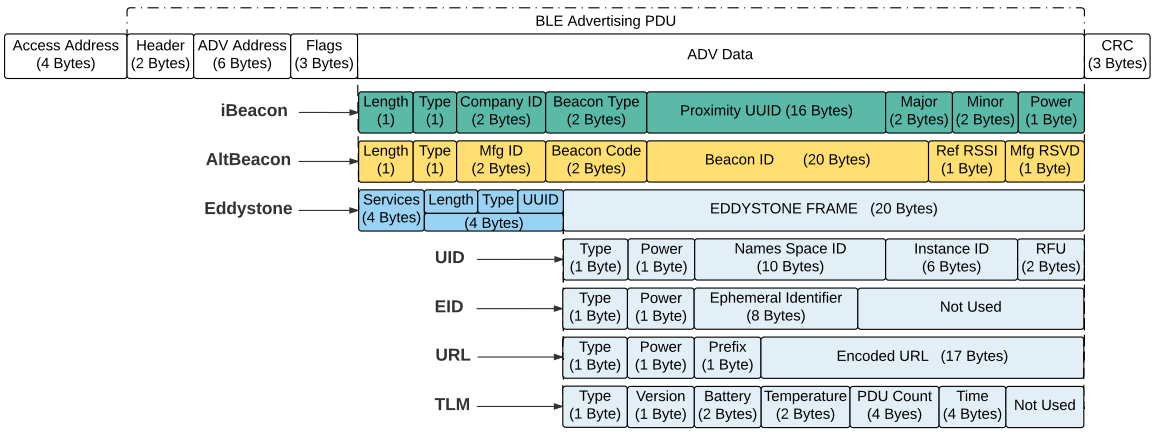


Figura 3.2: Estructura de los protocolos iBeacon, AltBeacon y Eddystone beacon.

utilizado ampliamente, excepto por algunos fabricantes tales como Gimbal [176], Swirl [177] and Shopkick [178], en consecuencia, no serán considerados como en el resto del manuscrito.

- Uniform resource locator (URL): transmiten una URL que apunta a un sitio web protegido Secure Socket Layer (SSL).
- Telemetry frame (TLM): marcos que transmiten información del beacon, como su nivel de batería o los valores de los sensores embebidos. Dado que esta tesis se centra en aplicaciones de telemetría, estos son los marcos más importantes.

En cuanto a los fabricantes de beacons para el mercado IoT, hay un gran número de ellos, que están compitiendo para el posicionamiento de sus productos en diferentes aplicaciones en términos de bajos precios, tamaños más pequeños y tiempo de vida de la batería. La tabla 3.2 compara las características de algunos de los beacons comerciales más populares (más dispositivos y una excelente visión general de la historia de la tecnología de los beacons se presenta en [179]). En dicha tabla se puede observar que la mayoría de los chipsets son fabricados por dos gigantes de la tecnlogía, Nordic Semiconductor y Texas Instruments (TI) y, en menor medida, por Bluegiga y Qualcomm (que adquirió CSR). En cuanto a los protocolos soportados, Eddystone y IBeacon prevalecen sobre las otras soluciones.

Tabla 3.2: Comparación de los beacons comerciales mas relevantes.

Beacons	**Fabricante del Chipset**	**Protocolos**	**Batería**	**Tiempo vida Batería (meses)**	**Costos (USD $)**
Estimote [?]	Nordic	iBeacon or Eddystone	CR2450	21.4	3 for $59.00
BKON [?]	Nordic	iBeacon or Eddystone	2 x AAA	28.9	$30.00
RadBeacon Dot [?]	Nordic	iBeacon, AltBeacon or Eddystone	CR2032	5.00	$14.00
Konkat [?]	Nordic	iBeacon or Eddystone	CR2477	24.3	3 for $60.00
OpenBeacon BL [?]	Nordic	Eddystone	CR2032	N/A	$25.00
BlueSense [?]	Bluegiga	iBeacon	CR2450	12.7	£19.99
BlueCats [?]	Bluegiga	iBeacon or Eddystone	2 x AA	23.1	3 for $89.00
RedBear Beacon B1 [?]	TI	iBeacon	2 x AAA	21.00	$21.00
SWIRL [?]	TI	iBeacon or Eddystone	4 x AA	72.00	N/A
Glimworm [?]	TI	iBeacon	CR2450	11.0	4 for €99.00
Gimbal [180]	Qualcomm	iBeacon or Eddystone	4 x AA	16.4	$5.00

Una lista mas extensa de beacons comerciales se pueden encontrar en [180] y [181]. En dichas listas Kontak.io se destaca como el líder en la categoría de hardware, seguido de Estimote, mientras Swirl lidera la categoría de plataformas de proximidad [121].

3.2.5. Applicaciones de Beacon

Los beacons han sido diseñados para la recogida de datos oportunista [118], principalmente para estimar la distancia de lectura y la proximidad, y para transmitir notificaciones. Gracias a estas capacidades los beacons han sido útiles en la venta al

por menor, entretenimiento, museos, o incluso en aeropuertos [182]. Por otra parte, la investigación se ha centrado en el análisis de la navegación y técnicas de seguimiento [183, 184], esquemas de posicionamiento en interiores [160, 185, 186] y notificaciones de proximidad [187, 188]. Otras aplicaciones están en la implementación de la red física para obtener posiciones [189] O para habilitar páginas web educativas [190].

Una extensa revisión de 100 casos de uso mediante beacons se presenta en la referencia [191]. La mayoría de las aplicaciones se centran en soluciones de marketing de proximidad. Este hecho indica que los beacons no están siendo plenamente explotados en términos de valor añadido en otras aplicaciones, como en el caso de aplicaciones de telemetría. Una de las razones es que, aunque el protocolo de Eddystone puede transmitir tramas de telemetría, que sólo contienen dos campos físicos de telemetría (por ejemplo, el estado de la batería y un valor de temperatura), lo que limita el número de sensores a se notificados. Uno de los pocos ejemplos de telemetría a base de baliza se presenta en la Referencia [192], donde el estado de una entrada on/off digital es transmitida.

Hay trabajos preliminares que encapsulan nuevos protocolos en un protocolo estándar de beacon. Por ejemplo, en la Referencia [193] se propone una técnica de intercalado que hace uso de múltiples anuncios BLE que identifican los dispositivos basados en el protocolo Altbeacon. Otra patente interesante es la referencia [194], donde se propone una técnica que permite la identificación de dispositivos conectados a iBeacons. Tal identificación se realiza mediante la transmisión de patrones sucesivos en diferentes niveles de potencia. Entonces, es posible identificar y calibrar el dispositivo conectado a través de su patrón de potencia.

El protocolo ligero presentado en este capítulo (Lightweight Protocol for Eddystone, LP4Sensors) se puede encapsular en los frames estándar de un Eddystone Beacon. El protocolo permite la transmisión de datos de telemetría de ilimitados sensores heterogéneos digitales y analógicos, limitado solamente por el pinout del hardware del beacon. El protocolo se inserta en las tramas del beacon Eddystone sin afectar el estándar de sólo 6 bytes. Además, es capaz de identificar los sensores y actuadores conectados, incluso incluyendo una información sobre ellos o su pinout. Además, el descubrimiento y el registro de transductores en un sistema IoT puede llevarse a cabo a través de un mecanismo de plug-and-play.

3.3. Protocolo ligero para sensores (LP4S - Lightweight Protocol for Sensors)

En esta sección se presenta LP4S, una familia de protocolos ligeros que pueden:

- Ser embebido en beacons BLE.

- Ser utilizado para enviar los datos recogidos por los sensores del beacon y recibir comandos para los actuadores.
- Garantizar la QoS requerida.
- Proporcionar bajo consumo de energía.
- Ser capaz de descubrir y describir el hardware integrado de forma automática mediante el uso de técnicas de plug-and-play.

Aunque el protocolo LP4S fue diseñado para ser utilizado principalmente para sistemas de telemetría, seguimiento, control y posicionamiento, su flexibilidad permite que se extiende a otras aplicaciones que requieren para comunicar dispositivos de recursos limitados.

El protocolo de LP4S la define tres tipos de frames:

- Configuración. Se utilizan para describir el hardware del beacon, permitiendo que se registre automáticamente en sistemas IoT a través de mecanismos de plug-and-play.
- Datos. Se utilizan para enviar los valores reales de los sensores y el estado de los actuadores, ya sean digitales o analógicos. También se utilizan para enviar comandos unidireccionales a los motes.
- Comandos Bidireccionales. Este tipo de frame es opcional y depende de los recursos y el tipo de beacon utilizado. Por ejemplo, este tipo de frame no tiene sentido en ciertas balizas que sólo funcionan en el modo fuera de línea, y que sólo envían la información en cada advertisement. Sin embargo, cuando se requiere una comunicación bidireccional, estos frames se pueden utilizar para enviar comandos al beacon, los cuales a su vez enviarán una confirmación de la orden recibida hacia el servidor, gateway o dispositivo que ha generado el comando.

El protocolo tiene que cumplir los tres requisitos siguientes para ser considerado ligero:

- costo de almacenamiento bajo o bajo footprint. El firmware desarrollado no puede ocupar mucha memoria y que debería ser posible subirlo a beacons con recursos limitados.
- Bajo coste computacional. Los frames tienen que ser cortos, lo que permite acelerar las tareas de cálculo en ambos extremos del enlace de comunicaciones.
- Bajo coste de la comunicación. El número de tramas a transmitir tiene que minimizarse tanto como sea posible para reducir el coste de la comunicación en términos de consumo de energía.

Dada la diversidad de los beacons IoT disponibles en la actualidad, que difieren en términos de restricciones de hardware y tecnologías de la comunicación, fue necesario crear una familia de protocolos que permitan cumplir con los requisitos detallados anteriormente. Para tal propósito, tres grupos de beacons se definieron en función de sus recursos:

- Beacon tipo-1. Están muy limitados en términos de recursos de hardware y la complejidad de protocolo (es decir, la cantidad de bytes y tramas que deben ser procesadas son realmente restringidas). Este es el caso de la mayoría de los beacons.
- Beacon tipo-2. Tienen más recursos de hardware que el tipo 1 y los beacons son capaces de manejar grandes frames. Este es el tipo beacons que ofrecen los datos del GATT a través de comunicaciones BLE.
- Beacon tipo-3. Estos son dispositivos IoT que, aunque alimentado por baterías, tienen más recursos de cómputo y de memoria que permiten correr firmwares más complejos y para el procesamiento de datos de tipo web, como archivos JSON (JavaScript Object Notation).

Por lo tanto, la familia LP4S está formada por tres sub-protocolos:

- LP4S-6: se utiliza para beacons Tipo-1.
- LP4S-X: se utiliza para beacons Tipo-2.
- LP4S-J: se utiliza para beacons Tipo-3.

Como será explicado en las siguientes secciones, cada uno de estos tres protocolos hace uso de un conjunto de frames que han sido diseñados en busca de un equilibrio entre la cantidad de bytes a transmitir, el número de frames que intervienen en las comunicaciones y la complejidad de dichos frames.

3.3.1. LP4S-Six (LP4S-6)

Este protocolo sólo hace uso de los frames de configuración y de datos, ya que fue diseñado desde su concepción para ser utilizado en beacons Eddystone. Sin embargo, el protocolo no se limita a un tipo de beacon específico, pudiendo ser encapsulado en otras beacons estándar existente o protocolos propietarios, e incluso en las comunicaciones serie entre los chips. Tenga en cuenta que la principal restricción de los beacon tipo-1 es el tamaño de las tramas de protocolo, por lo que fue necesario diseñar un protocolo que cumpla con los requisitos deseados en sólo 6 bytes.

La figura 3.3 muestra la estructura de este protocolo. Se puede observar que el protocolo está formado por tres grupos de dos bytes. Cada grupo tiene más de una funcionalidad dependiendo del tipo de la información gestionada (es decir, descripción, configuración o ciertos datos). El primer grupo de dos bytes (Descripción Global), contiene informaciones generales sobre el beacon y la información que se envía. Tal información se divide en cuatro nibbles que incluyen el modo de trabajo, tipo de transductor incorporado, tipo de señal transmitida y el tipo de interfaz. Por lo tanto, es posible indicar si la información enviada está relacionada con la configuración o los datos de un sensor específico, si es digital o analógico, o si los datos transmitidos se relacionan con entradas o salidas. Como cada nodo permite el uso de hasta 16 descripciones diferentes (algunas de las cuales aún no se han definido), se puede afirmar que el protocolo es flexible, escalable y está abierto a futuras mejoras y personalización para nuevas aplicaciones IoT.

El segundo grupo de dos bytes (Descripción del hardware/Valor de datos), su contenido depende de la información del grupo anterior:

- Cuando el modo de trabajo en la Descripción Global indica que una trama configuración está siendo enviado, entonces el primer byte contiene el número total de sensores y actuadores activos en el beacon. Entre otras funcionalidades,

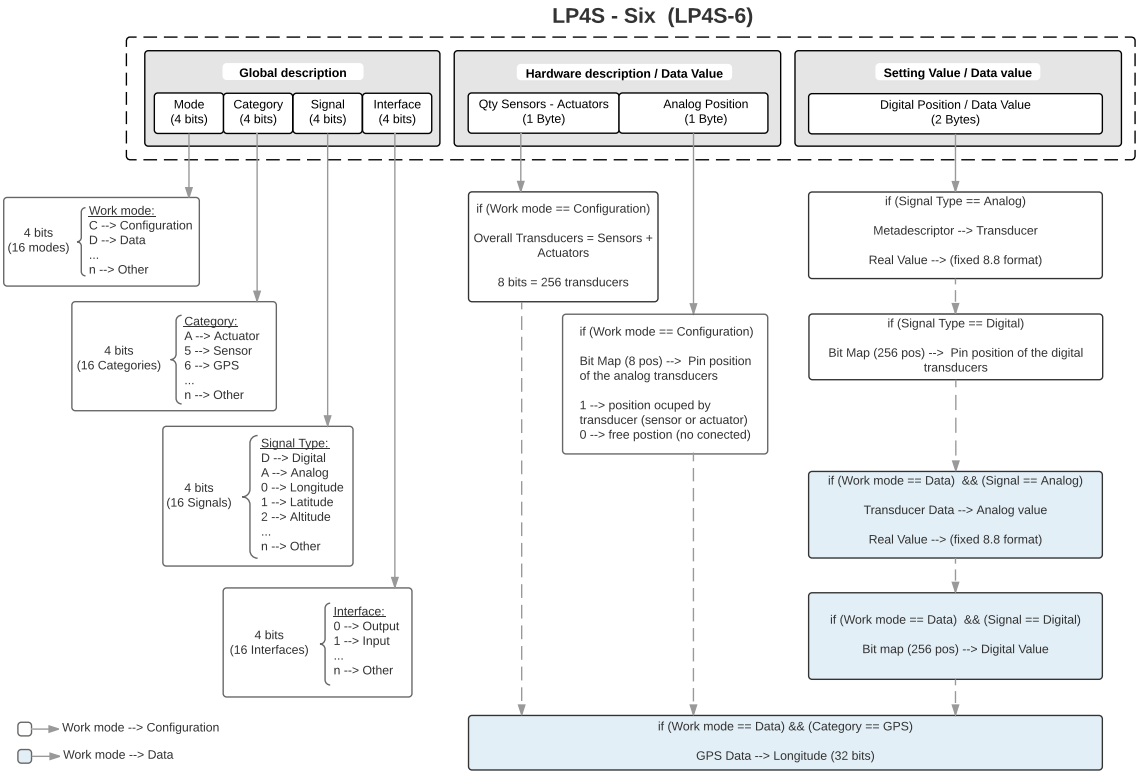


Figura 3.3: Estructura interna del protocolo LP4S-6.

esta actividad de campo permite a usuarios remotos indicar si han recibido toda la información de la baliza. Además, en este modo de trabajo, el segundo byte es un mapa de 16 bits que indica (cuando es igual a "1") cuando un transductor está conectado y operando.

- En el modo de datos de ambos bytes contienen el valor leído desde el transductor especificado en la Descripción Global.

El tercer grupo de dos bytes (Valor de ajuste / valor de datos) también depende del contenido de la descripción global:

- En el caso del frame de configuración, este campo indica la posición y el tipo de datos de transductor (es decir, analógica o digital).
- Cuando se transmite una trama de datos, el valor del sensor está indicado con una precisión de 16 bits. Si el transductor incorporado es analógico, entonces los dos bytes contienen un valor real en un formato de punto fijo-8,8. En el caso de sensores digitales, la información se proporciona en forma de un mapa de 16 bits. Si lo que se informa es una posición GPS, a continuación, un "1" indica que el transductor está activo. Por último, si lo que se informa es un valor digital, un mapa de bits se dará en la que se establece el bit relacionado con la posición del sensor digital a "0" o "1", dependiendo de su estado.

Para una mejor comprensión del protocolo, varios ejemplos se muestran en la figura 3.4 . El caso (a) representa un frame de configuración que indica que hay un total de 5 transductores, dos de los cuales corresponden con sensores digitales localizados en las posiciones 0 y 1 del mapa (0x0003 en hexadecimal es igual a 0000 0000 0000 0011 en binario). En el ejemplo (b) se indica que hay una salida digital situado en la última posición de mapa de bits (0x8000, 1000 0000 0000 0000). Ejemplos (c) y (d) representan frames de configuración donde se utilizan dos sensores conectados a las entradas analógicas situados en la quinta (0001 0000) y sexta posición (0010 0000). Además, en estos dos últimos ejemplos, los meta-descriptores asociados (7,00 y 1,00) representan índices de una tabla que contiene una descripción más detallada de los transductores. Por ejemplo, el sensor identificado por el meta-descriptor 7.00 podría representar un sensor de temperatura dada en °C, mientras que el identificado por 1,00 podría ser un sensor de humedad relativa cuya unidad de medida es un porcentaje.

Los ejemplos desde (e) hasta (j) muestran donde se envía la información recogida por los sensores descritos anteriormente en (a) a (d). En (e) se indica que, de los dos sensores digitales definido en (a), la entrada digital primero se establece en "0" y la segunda, un "1" lógico (por ejemplo, debido a la activación de un sensor PIR o cuando una puerta

este abierta). El ejemplo (f) indica el estado de una salida digital que se establece en "1". Los casos de uso (g) y (h) emiten valores en tiempo real de dos entradas analógicas, que, de acuerdo con las tramas configuración enviados anteriormente en (c) y (d), y las suposiciones indicadas acerca de su tipo de sensor, se indicaría una temperatura de 23,15 °C y una humedad relativa del 85,07 %. Por último, (i) y (j) muestran la posición geográfica actual del beacon, que estaría ubicada en las coordenadas (43.33640, - 8.412977).

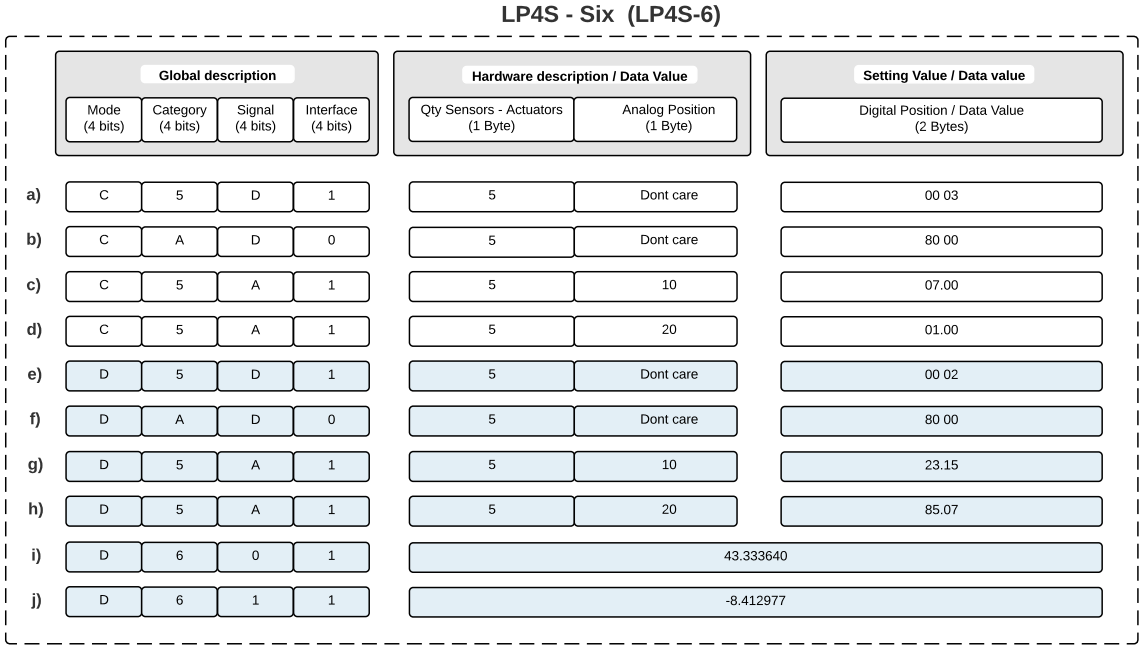


Figura 3.4: Ejemplo de uso del protocolo LP4S-6. (a - d) Frames de Configuración frames: sensores y actuadores digitales y analógicos, (e - h) Frames con datos del sensor y actuadores digitales y analógicos, (i - j) Frames de Localización frames: longitud y latitud.

3.3.1.1. LP4S-Extended (LP4S-X)

Este protocolo es una variación del protocolo LP4S-6 que se ha modificado para evitar la restricción de tener sólo 6 bytes por trama. Por lo tanto, constituye una extensión del protocolo anterior (de ahí su nombre). Aunque no hay restricciones en el número de bytes, sólo dos de los diez frames LP4S-X en realidad tienen más de 6 bytes (algunos de ellos contienen sólo 4 bytes). Este protocolo hace uso de los tres tipos básicos de tramas representadas en diferentes colores en la figura 3.5, los cuales están relacionados con configuración, el intercambio de datos y comandos bidireccionales.

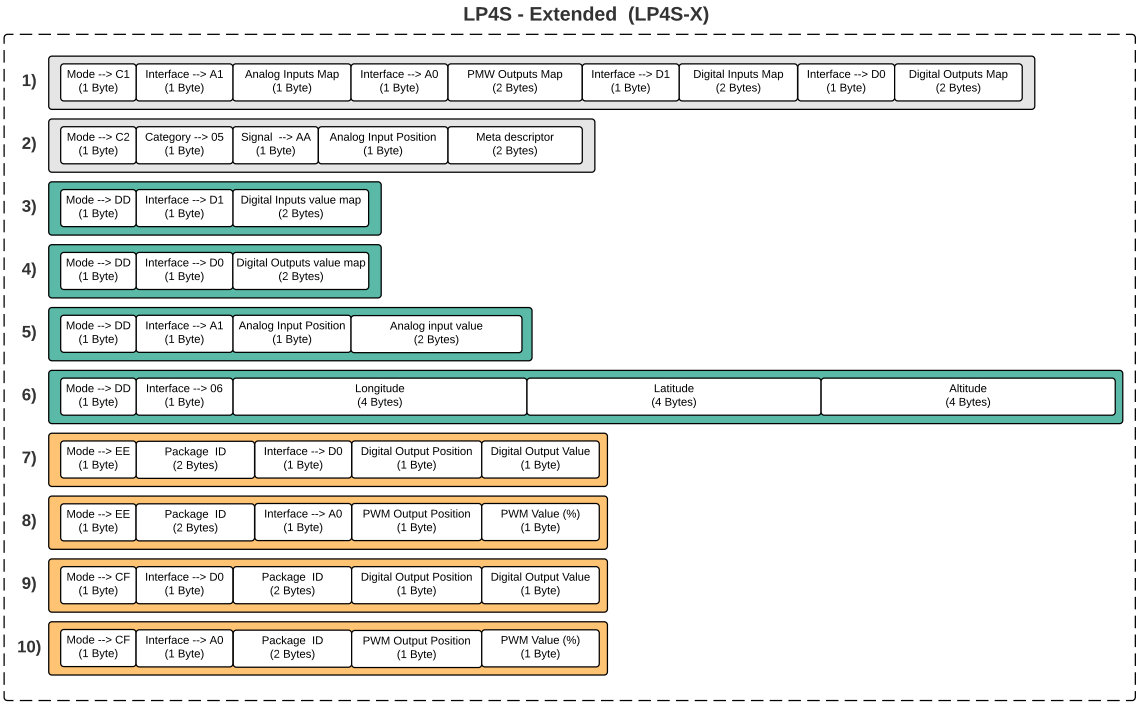


Figura 3.5: Estructura interna del protocolo LP4S-Extended (LP4S-X) (1-2) Frames de Configuration, (3-5) Frames de lecturas para digitales, analógicos y datos de localización, (7-8) Frames para comandos digitales y analógicos, (9-10) Frames de confirmación

Cada uno de los frames comienzan con un byte que permite identificar el tipo de trama: C1 y C2 representan los frames de configuración, DD son tramas de datos, EE son tramas de comandos y CF frames de configuración.

Como se puede observar, existen dos frames de configuración en este protocolo, C1 y C2. El primer tipo, cuya estructura se muestra en el frame (1) en la figura 3.5, Se utiliza para describir la posición de todos los transductores de entrada y salida (tanto analógicas como digitales) a través de mapas de bits, de manera similar a LP4S-6. En figura 3.5, en (2), se muestra la estructura del frame de configuración C2, que se utilizan para describir cada transductor. Por lo tanto, se enviarán tantos frames C2 como transductores con requerimientos de descripción estén conectados al beacon. Aunque C2 se utiliza normalmente para sensores analógicos, pueden ser utilizados para describir cualquier otro transductor que requiera una descripción detallada.

Los frames (3) y (4) de la figura 3.5informan el estado de las entradas y salidas digitales mediante el uso de mapas de bits. En el caso de sensores analógicos, también es necesario enviar una trama como (5) con el fin de indicar los valores detectados. En cuanto al frame (6), se incrusta en un solo frame toda la información de geolocalización (longitud, latitud y altitud) de una posición.

Los frames (7) y (8) se utilizan para enviar comandos a los actuadores del beacon, cuya respuesta puede ser controlada (por ejemplo, relés, LEDs o controladores PWM). Puesto que tales marcos no utilizan mapas de bits, una trama debe ser enviada para cada salida PWM digital a ser modificada.

Por último, (9) y (10) representan las tramas que se envían por el beacon al servidor con el fin de confirmar que se ha recibido un comando. Tales frames incluyen un campo ID, único para cada mensaje, que ayuda al servidor para identificar el comando reconocido. Estos frames también se utilizan para enviar el estado actual de una salida modificada previamente, lo que permite establecer alarmas o mecanismos de retransmisión en el lado del servidor.

3.3.2. LP4S-JSON

El tercer y último protocolo de la familia LP4S ha sido diseñado para beacons que tienen acceso a una cantidad relevante de recursos. En este tipo de dispositivos es posible implementar protocolos como el COAP, que normalmente intercambian información en formato Java Script Object Notation (JSON). En LP4S-JSON hay tres tipos de tramas:

- LP4S - Configuration (LP4S - JC): Frame de Configuración.
- LP4S - Read (LP4S - JR): Frame de Datos de sensores.
- LP4S - Write (LP4S - JW) Frame para Comandos / Confirmación.

Cada uno de estos frames contienen datos en formato JSON que hacen uso de las etiquetas mostradas en la tabla 3.3 con el fin de crear una representación semántica del beacon. Tal representación permite la descripción de su hardware y para descubrir y auto-registro automático de los transductores a través de mecanismos de plug-and-play. Algunas de las etiquetas JSON en la tabla 3.3 indican el nombre del beacon, su dirección MAC, diferentes meta-descriptores o un mapa de bits de los sensores digitales. Hay incluso una etiqueta para notificar a un escáner BLE cuando se pulsa un botón.

3.3.2.1. LP4S - JSON Configuración (LP4S-JC)

Al igual que en los otros protocolos de la familia LP4S, los formatos de configuración diseñados permiten la descripción del hardware existente y facilitar el descubrimiento fácil y registro del hardware de aplicaciones de la IO. Gracias a las etiquetas de la tabla 3.3 es posible llevar a cabo tales tareas de detección y registro con una descripción semántica que puede ser adaptado a la Wot.

Varios ejemplos de uso del protocolo LP4S-JC se muestran en la figura 3.6. Tenga en cuenta que, con el fin de mantener la ligereza del protocolo, es posible enviar un

micro-JSON por parámetro de configuración (es decir, en la figura 3.6 (b) hasta (g) en lugar de enviar todos los parámetros (como en la figura 3.6 (a)) en un sola JSON. Es importante señalar que cada micro-JSON siempre incluye la dirección MAC del beacon como un parámetro, que garantiza la identificación del dispositivo. Por último, en (h) se ilustra un ejemplo de un parámetro de tipo mixto que representa una entrada ('in') de un sensor analógico ('a') situado en la segunda posición ('2' en decimal o "0000 0000 0000 0010" en binario) de un mapa de bits. Esta clase de tipos se utilizan para reducir el número de etiquetas JSON y mantener la ligereza del protocolo en ciertas aplicaciones.

3.3.2.2. LP4S - JSON Read (LP4S-JR)

Este protocolo se utiliza para representar los datos de los frames que el beacon envía cuando el valor de algunos de sus sensores ha cambiado. Sin embargo, el firmware del beacon podría ser modificado para utilizar otros criterios para desencadenar el envío de paquetes.

LP4S-JR hace uso de micro-JSONs cuya estructura se muestra en la figura 3.7. En dicha figura, en (a) se se representa un caso en el que se emite una temperatura de 23,5 °C del sensor situado en la posición 2. En el ejemplo (b) se muestra el estado actual de todas las entradas y salidas digitales (tenga en cuenta que se utilizó la etiqueta especial "todos los pines"). En cuanto al ejemplo (c), se ilustra una notificación a un servidor cuando se pulsó el botón ubicado en la primera posición. Por último, en (d) se muestra el JSON para enviar las coordenadas de geolocalización del beacon.

Tabla 3.3: Principales etiquetas usadas en los JSONs.

Etiqueta	Descripción	Valor de ejemplo
MAC	MAC del beacon	"00:11:22:33:44:55"
Name	Nombre de beacon	"Z1-T1"
Manufacturer	Código del Fabricante	"070000000001"
Location	Longitud, Latitud y Altitud	"(00,-00, 00)"
Description	Fabricante, distribuidor, tecnología, otros	"nRF51-Dk"
Qty_all	Total de transductores (sensores + actuatores)	"8"
Metad	Meta descriptor	"1.0"
Interface	Interface + señal + posición	ïna7"
Button	Botón presionado	"1"
Pos_dig_in	Mapa de posiciones (Entradas Digitales)	"3"
Pos_dig_out	Mapa de posiciones (Salidas Digitales)	"128"
Pos_ana_in	Mapa de posiciones (Entradas Analógicas)	"5"
Pos_pwm_out	Mapa de posiciones (Salidas PWM)	"136"
Package_id	Identificador de paquete	"1005"
Value	Valor del dato (Lectura o Escritura)	"23.5"

3.3.2.3. LP4S - JSON Write (LP4S-JW)

La estructura de los mensajes comandos y confirmación se muestran a través de los ejemplos de la figura 3.8.

En el caso de (a), se representa un comando que activa un actuador digital (por ejemplo, un relé o un LED) que está conectado en la segunda posición ('outd2') del beacon. El ejemplo (b) muestra un comando que cambia la salida PWM situada en la tercera posición ('outa3') a 44 %. Por último, el ejemplo (c) contiene el mensaje de confirmación que sería enviado por un beacon después de recibir el comando mostrado

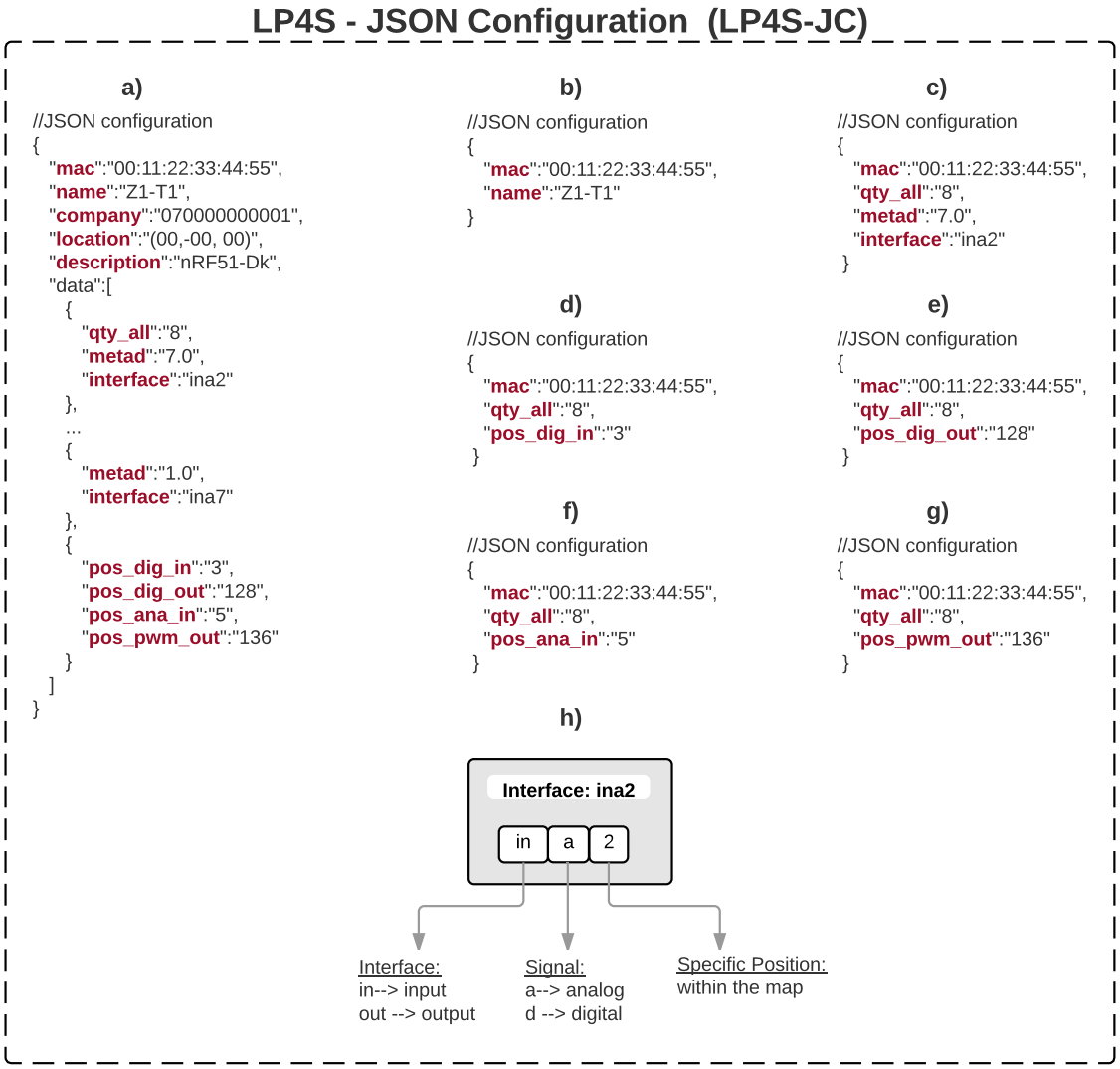


Figura 3.6: LP4S-JC mensajes que muestran JSONs que incluyen todas la etiquetas (a), un subconjunto de las etiquetas (b-g), y una parámetro mixto (h).

en (a). Tenga en cuenta que, en este último ejemplo (c), el valor de "package-id" tiene que coincidir con la de (a). Sin embargo, observe que en (c) el estado de 'valor' es '0', mientras que en (a) se indicó que tenía que ser puesto a '1', lo que indica que el comando no se ha procesado o se procesó erróneamente. En tal caso, el servidor reaccionaría de acuerdo con la lógica programada (por ejemplo, que podría retransmitir automáticamente el comando).

3.4. Implementación del protocolo LP4S en un beacon

La familia de protocolos LP4S puede ser implementado en cualquier beacon o mote cuyo hardware siga la arquitectura ilustrada en la figura 2.1. Tenga en cuenta que este tipo de arquitectura es sólo una referencia, por lo que puede existir cierta diferencia en términos de hardware, pero el protocolo todavía puede ser compatible. Por ejemplo, el subsistema de comunicaciones podría estar integrado en el chipset del beacon System on a chip (SoC) o en un módulo externo.

Esta sección está dirigida específicamente a la aplicación del protocolo LP4S-6 en un beacon de recursos limitados como el Eddystone. Este beacon transmite la cierta

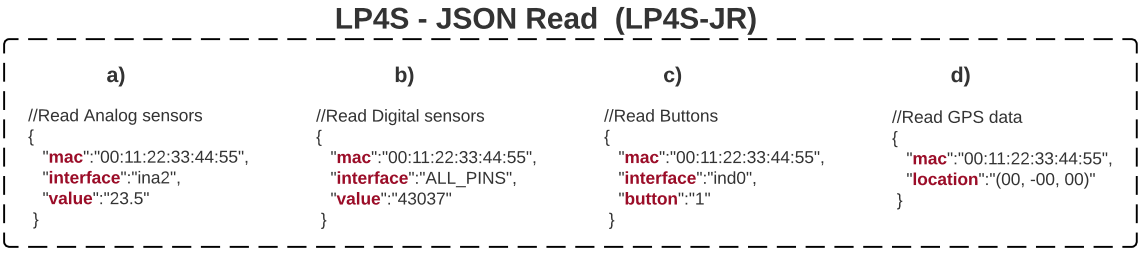


Figura 3.7: Ejemplos de uso del protocolo LP4S-JR: (a) dato analógico, (b-c) dato digital y (d) dato de geolocalización.

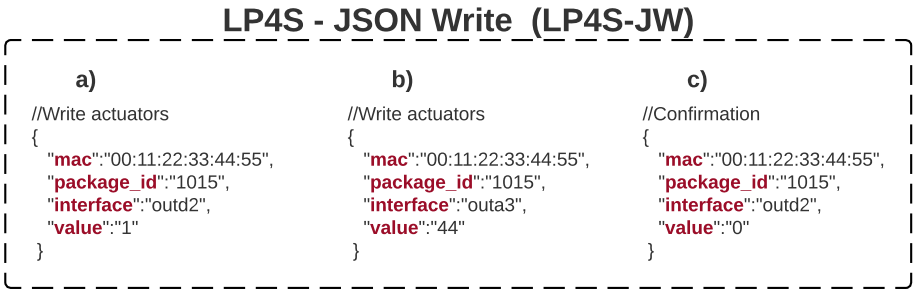


Figura 3.8: Ejemplos de uso del LP4S-JW. (a) comando digital, (b) comando analógico, (c) mensaje de confirmación.

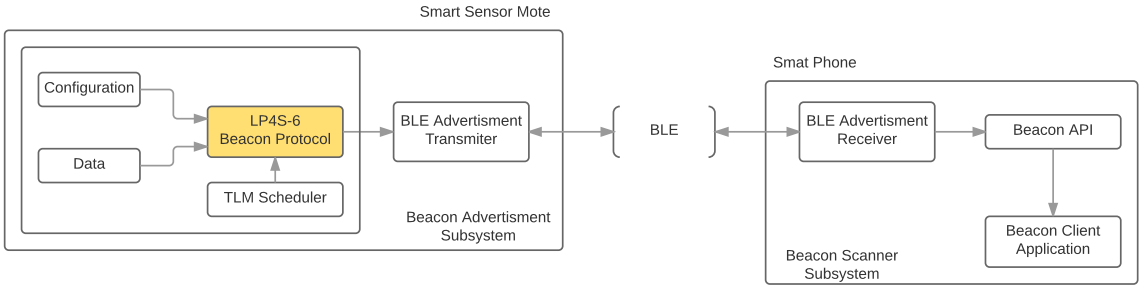


Figura 3.9: Arquitectura de comunicaciones de un beacon genérico.

información periódicamente con cada advertisement BLE. La figura 3.9 muestra la arquitectura de comunicaciones de un sistema de beacons genérico y un ejemplo de cómo el protocolo LP4S-6 se inserta en el planificador de frames Telemetry frame (TLM). Cualquier aplicación que se ejecuta en un escáner de beacons (por ejemplo, en un teléfono inteligente o en un Session Bordere Controller (SBC)) que implementa la Application Programming Interface (API) propuesta en este documento es capaz de leer los dos frames convencionales Eddystone y la información de telemetría encapsulada por el protocolo LP4S-6. Además, dado que un escáner normal Eddystone no se verá afectado por las modificaciones realizadas por el LP4S-6, que continuará leyendo beacons que hacen uso de ambos protocolos. Por lo tanto, un beacon que implementa el protocolo LP4S-6 puede trabajar en una red de difusión de beacons con la presencia de múltiples y diferentes escáneres de beacons de diferentes fabricantes y diferentes aplicaciones sin interferencia entre ellos.

Antes de detallar la implementación del protocolo, es importante observar que el firmware diseñado difunde tres de los cuatro frames del Eddystone (UID, URL y TLM) en un orden aleatorio y de acuerdo con una programación TLM, como se representa en la figura 3.10 para ilustrar el esquema de transmisión. En estos tres ejemplos de posibles secuencias de tramas, se puede observar que las tramas se envían cíclicamente, (es decir, una vez que la última trama fue enviada). El ciclo se inicia por el primer frame enviado de acuerdo con la secuencia establecida, que depende del número de sensores integrados y de la disponibilidad de datos leídos.

3.4.1. LP4S-6 en un beacon Eddystone

El protocolo LP4S-6 fue diseñado para ser insertado en protocolos de tamaño restringidos, tales como Eddystone, y que también utilizan tamaños de frames preestablecidos. Específicamente, en cada trama Eddystone TLM es posible encontrar 6 bytes que no se utilizan actualmente por la especificación Eddystone, donde el protocolo LP4S-6 se ajusta perfectamente. De esta manera, es posible adaptar los beacons Eddystone

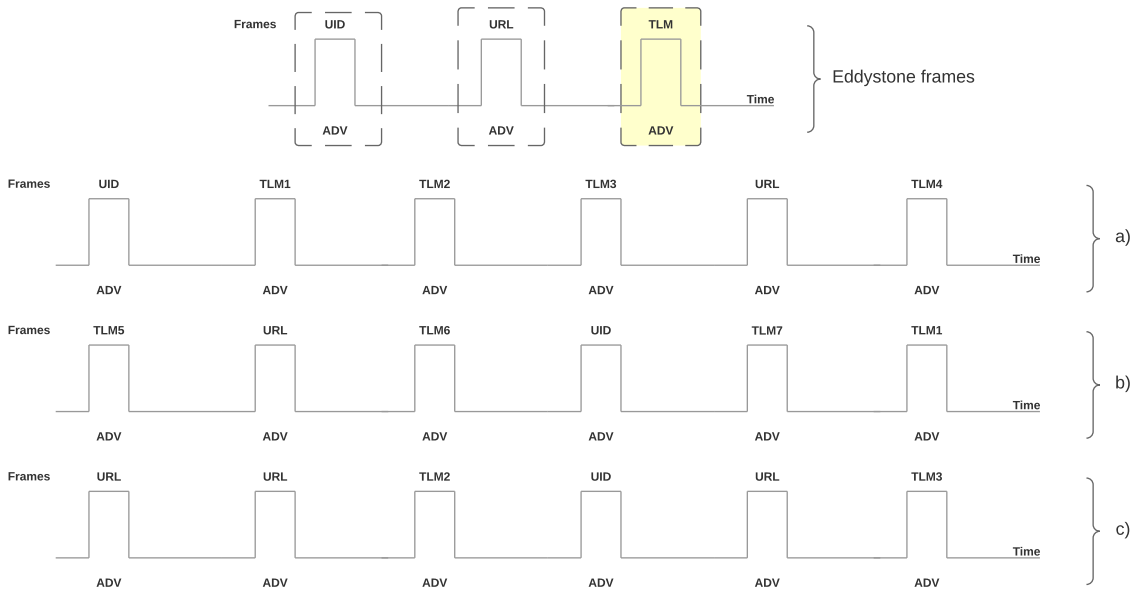


Figura 3.10: Ejemplos de secuencia de frames de un Eddystone.

tradicionales para ser utilizados en aplicaciones de telemetría que no requieren una conexión para su funcionamiento, lo que también es útil en ciertas aplicaciones IoT.

En figura 3.11 se muestra la ubicación precisa de los frames Eddystone TLM donde se inserta el protocolo LP4S-6. El espacio que aloja el protocolo propuesto se dividió en tres pares de bytes (Extra 1, Extra 2 y Extra 3).

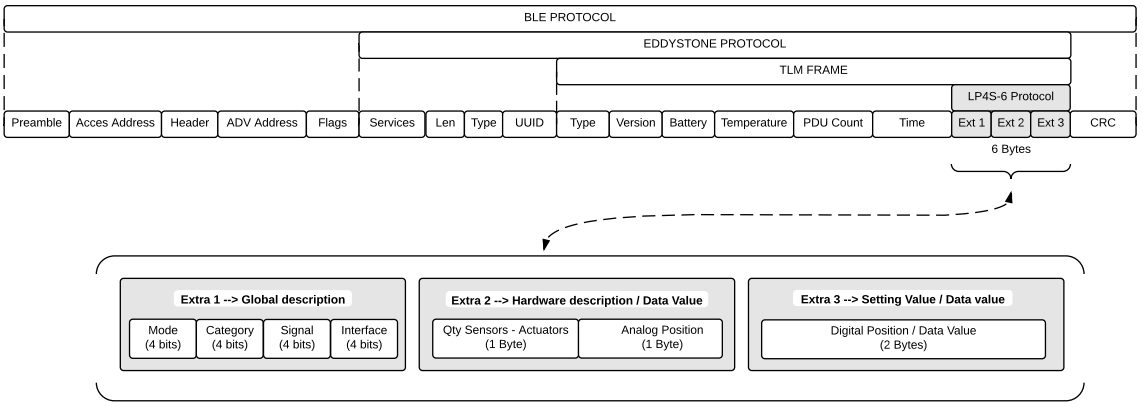


Figura 3.11: LP4S-6 insertado dentro de un frame TLM.

3.4.2. Diagramas de flujo

Esta sección describe los diagramas de flujo necesarios para implementar aplicaciones de telemetría en beacons comerciales, tanto que soporte el protocolo Eddystone y la LP4S-6, que añade el valor de ser capaz de utilizar sensores ilimitados (obviamente, tal límite depende de la cantidad de pines de entrada/salida analógicos y digitales disponibles en el hardware seleccionado).

La figura 3.12 (a) muestra el diagrama de flujo del programa principal de un beacon convencional, cuyas tareas principales son la configuración del hardware, y la inicialización de los parámetros y servicios del beacon para llevar a cabo comunicaciones BLE. Después de la inicialización, el beacon espera por interrupciones generadas por un temporizador para comenzar a intercambiar tramas siguiendo el flujo mostrado en la figura 3.12-b. Cuando llega el momento de transmitir una trama de TLM, los diferentes campos de una trama Eddystone estándar y los bytes adicionales del protocolo LP4S-6 se actualizan de acuerdo con el flujo mostrado en la figura 3.12-c. En el diagrama de flujo dos parámetros específicos se envían en cada frame TLM (el nivel de la batería

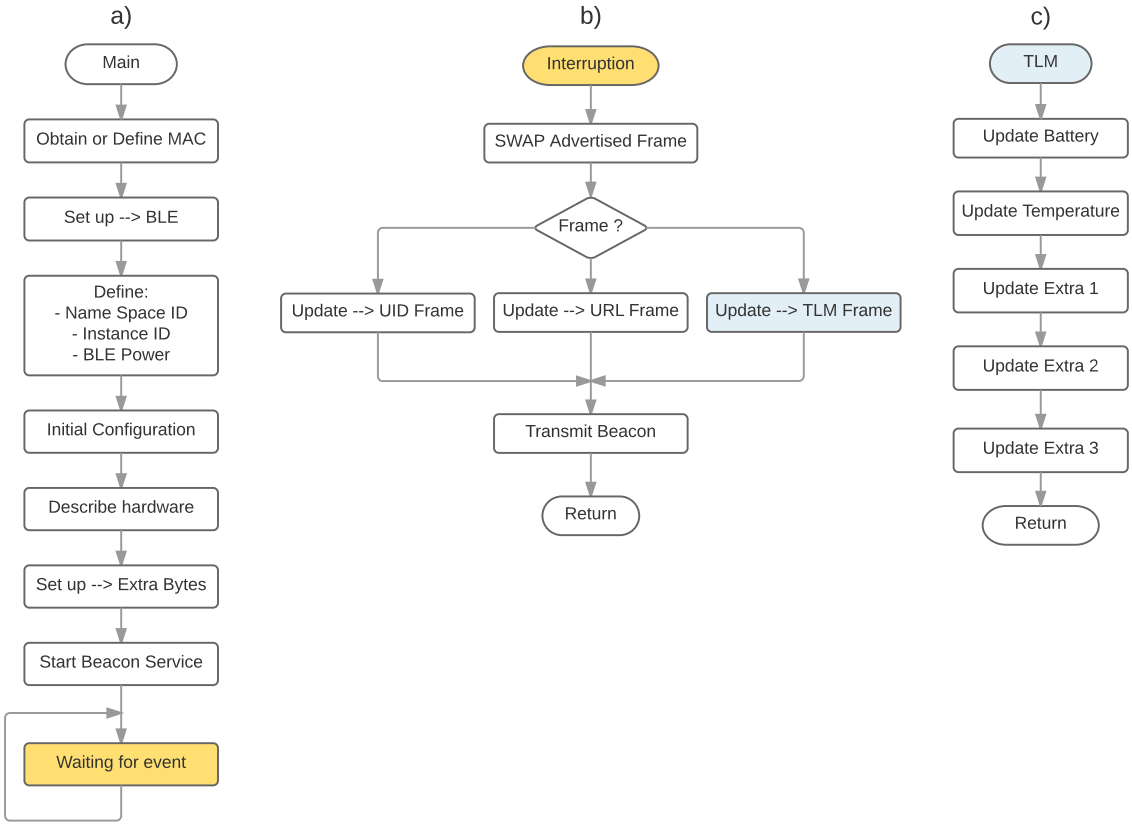


Figura 3.12: Diagramas de flujo de los beacons. (a) Principal (Main), (b) subrutina de interrupción del Advertisement (ISR - Interrupt Service Routine), y (c) flujo TLM.

y la temperatura ambiente), que se actualiza periódicamente por un hilo de ejecución paralelo y se almacena en un área de memoria compartida.

Como se explica en la Sección 3.3.1, los campos del protocolo LP4S-6 son multifuncionales, lo que es necesario para implementar un algoritmo que controla la información requerida en cada uno de los tipos de trama a transmitir. Figura 3.13 muestra el diagrama de flujo que ilustra cómo se implementa el byte extra-1. Tal byte corresponde al campo de descripción global del protocolo LP4S-6, por lo que el algoritmo permite el llenado de los campos relacionados con el modo (configuración o datos), Categoría (sensor o actuador), señal (analógica o digital) y la interfaz (entrada o salida).

Los dos bytes correspondientes a extra-2, que representan la Descripción del hardware / Valor de datos de campo, se llenaron de acuerdo con el diagrama de flujo mostrado en la figura 3.14. Este campo se utiliza principalmente para la configuración. Sólo en el caso de tener que indicar un punto de geolocalización, Extra-2 se unirán a Extra-3 para proporcionar un campo de datos de 32 bits. En Modo configuración, el primer byte de extra-2 contiene la cantidad total de los transductores, mientras que el segundo byte indica la posición de los transductores a través de un mapa de bits: si un bit es cero, indica que en esa posición no hay transductores activos.

Finalmente, la figura 3.15 muestra el diagrama de flujo seguido para llenar el byte Extra-3, ya sea con valores de los sensores, meta-descriptores o, como se mencionó anteriormente, uniéndose adicional a Extra-2 para proporcionar un campo de 32 bits para las coordenadas de geolocalización. La información digital se proporciona a través de un mapa de bits, mientras que los datos analógicos hacen uso de un formato de punto fijo 8.8 que también se utiliza en los tramas estándar Eddystone TLM.

3.5. Experimentos

En esta sección se compara el rendimiento de un beacon que hace uso ya sea del protocolo LP4S-6 o de un servidor GATT. En aras de la brevedad, sólo se evaluó LP4S-6, que es el protocolo más restrictivo de la familia LP4S. La comparación se llevó a cabo en términos de latencia y consumo de energía, en ocho escenarios diferentes: antes que la aplicación Android haga un çharacteristic connectçon el beacon, después de llevar a cabo una conexión de este tipo, y para Eddystone modificado con los protocolos propuestos para intervalos de beacons de 100 ms, 250 ms, 500 ms, 1000 ms, 2000 ms y 5000 ms. Para todos los intervalos el escenario físico fue idéntico, existiendo línea de vista entre el beacon y el teléfono inteligente, a una distancia de 3m entre ambos. El impacto de la latencia de trasmisión a esa distancia es despreciable (el tiempo total (round-trip) es de alrededor de $20\,ns$), y aún para 100 m (máxima distancia teórica para BLE), el round-trip de menos de $1\,\mu s$, mientras que las mediciones de latencias

obtenidas fueron del orden de segundos. Por otra parte, la distancia no es relevante para las mediciones de consumo de energía, ya que la fuente de poder es fija (no se utilizaron esquemas de transmisión de potencia adaptativo).

3.5.1. Configuración de los experimentos

La figura 3.16 muestra los elementos que intervienen en los experimentos: un kit de desarrollo nRF51 (nRF51-DK) que actúa como beacon BLE, un Nordic Power Profiler Kit (PPK) que se enchufa en el kit de desarrollo para medir el consumo de corriente, un teléfono Android y un ordenador portátil ejecutando Android Studio y Nordic Power Profiler software kit. Las principales especificaciones de todos estos elementos se describen en la tabla 3.4.

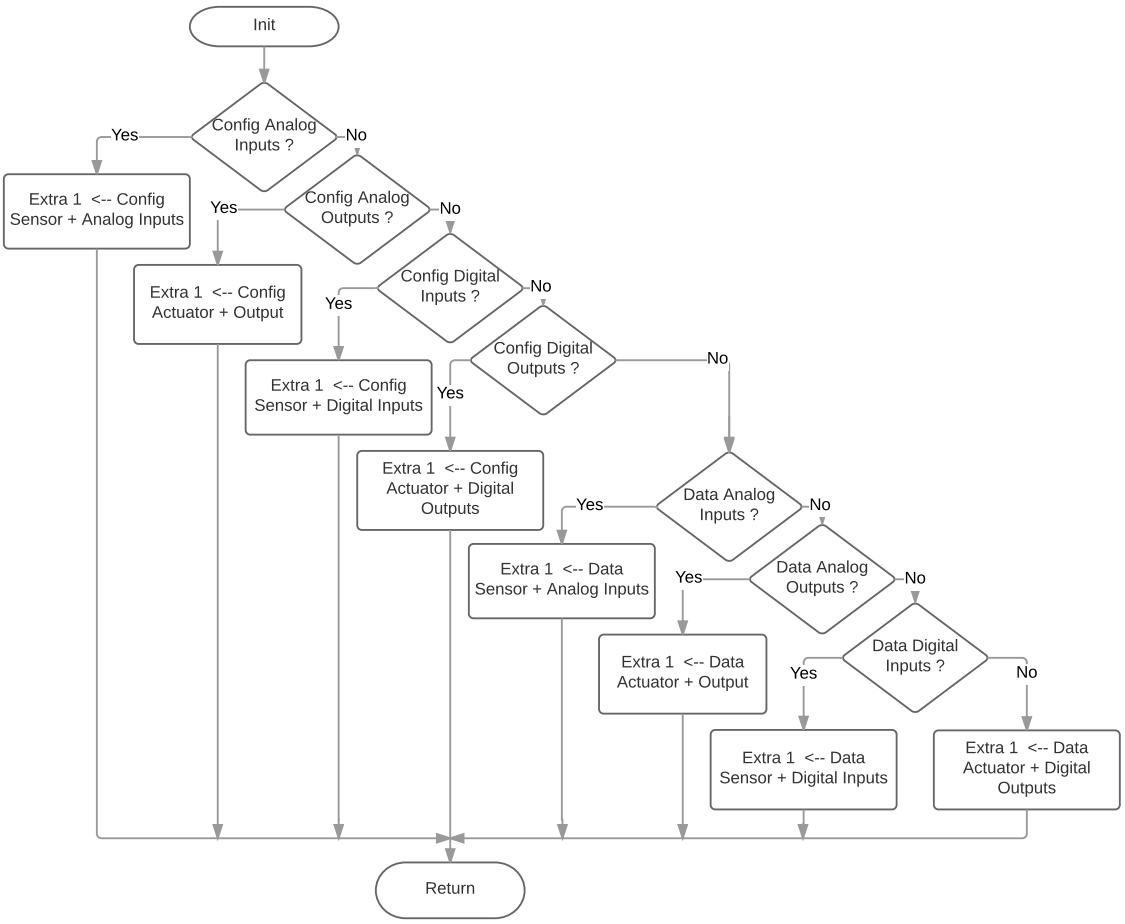


Figura 3.13: Diagrama de flujo para Extra-1.

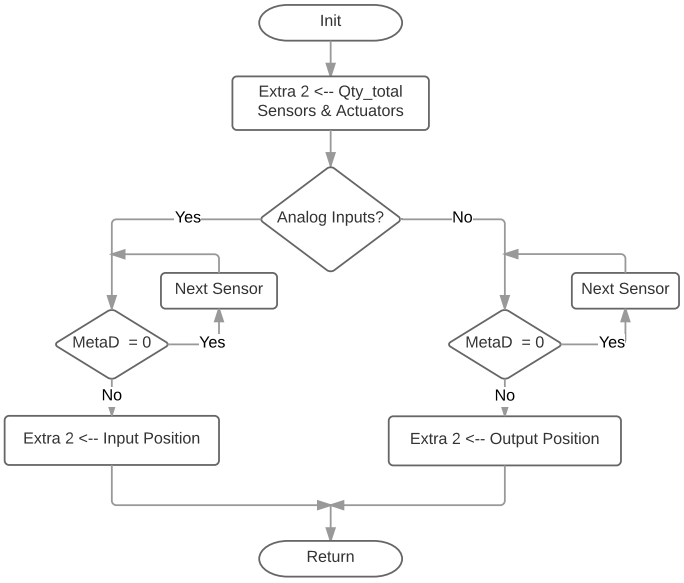


Figura 3.14: Diagrama de flujo para Extra-2.

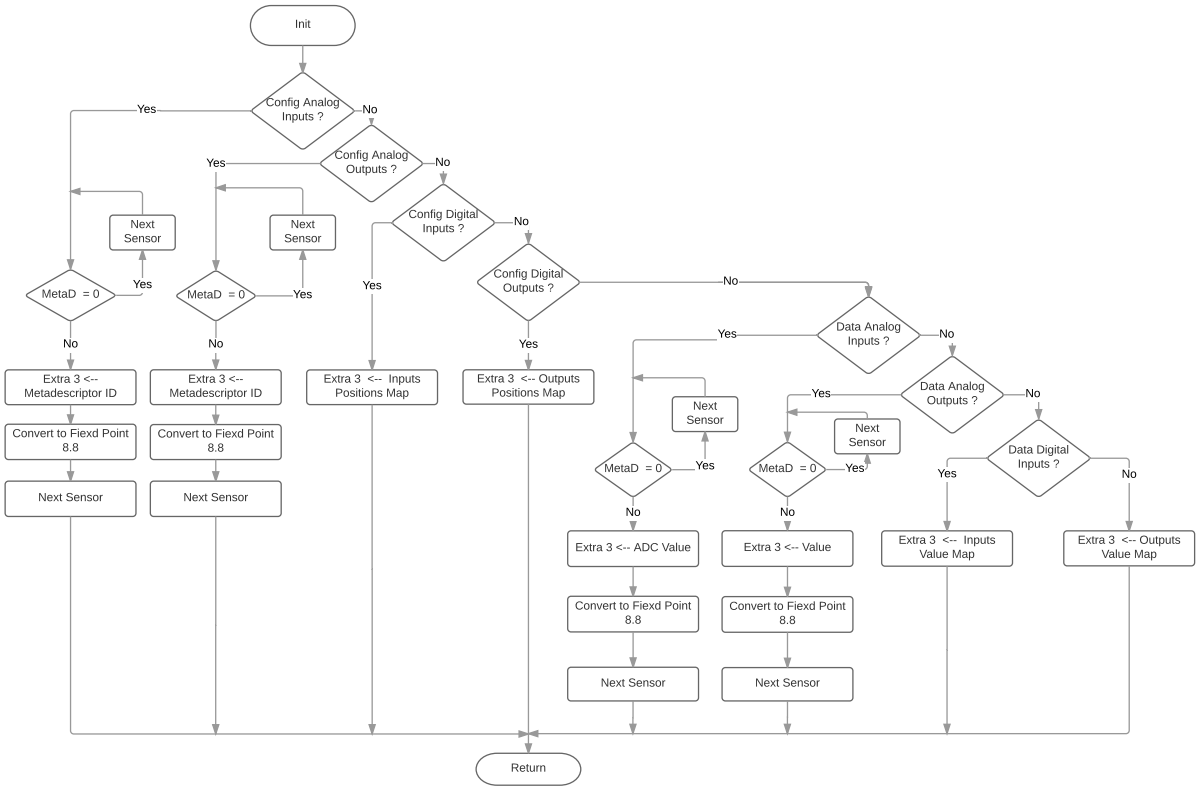


Figura 3.15: Diagrama de flujo para Extra-3.

Tabla 3.4: Especificaciones principales de los elementos del experimento.

Elementos usados	Especificaciones principales
Laptop	Lenovo 80E502A5SP G50-80: 15.6", 16 GB RAM, 1 TB HDD Intel® Core™i7-5500U CPU @ 2.40 GHz OS version: Windows 10-Pro x 64 bits
Android Studio 2.3.3	AI-162.4069837, built on June 6, 2017 JRE: 1.8.0.112-release-b06 amd64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Nordic nRF51 development kit (nRF51-DK)	SoC: nRF51822, 2.4 GHz multi-protocol device, 32-bit ARM® Cortex™M0 CPU with 256 kB/128 kB flash + 32 kB/16 kB RAM
Nordic PPK (Power Profiler Kit)	Board: PCA63511 Software: nRF6707-SW v1.1.0 Python version: 2.7.13, Pyside: 1.2.4, Pyqtgraph: 0.10.0, Numpy: 1.13.1, Pynrfjprog: 9.6.0
Smartphone	Samsung Galaxy S-8, 4 GB RAM, 64 GB (UFS 2.1) ROM Model: SM-G950F Android version: 7.0 (Nougat) Processor: Exynos 8895, 2.3 GHz Quad + 1.7 GHz Quad, 8 Cores (Octa-Core)

3.5.2. Mediciones de Latencias

La latencia es clave en la recogida de datos de los sensores en ciertas aplicaciones y también es una medida útil desde la experiencia del usuario. En los experimentos

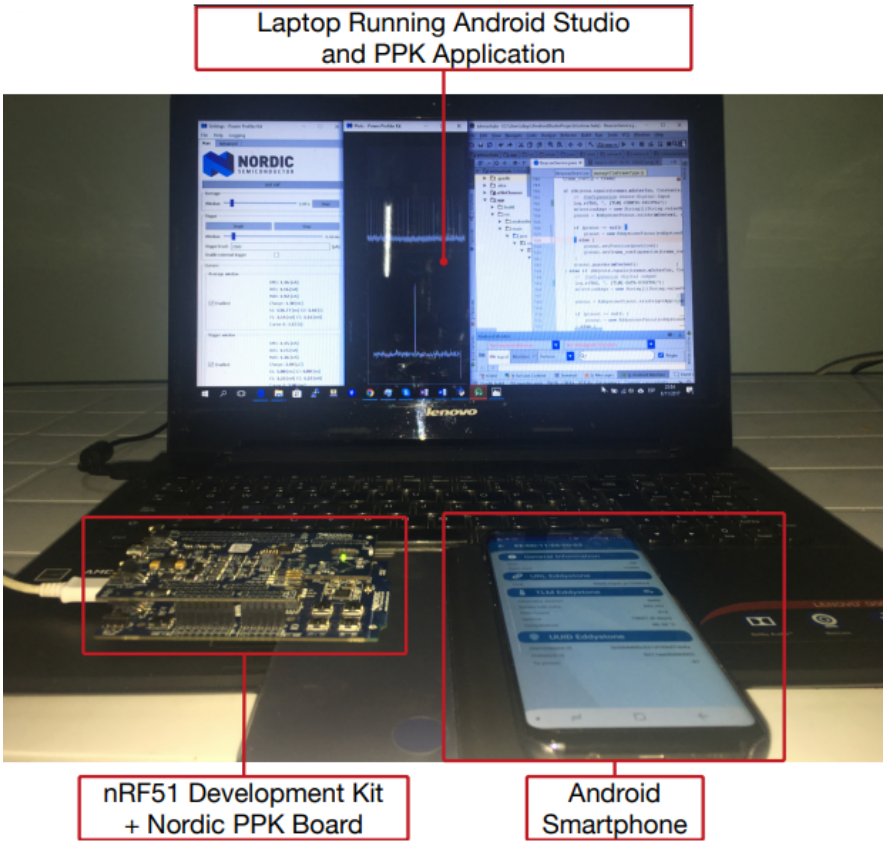


Figura 3.16: Configuración de los experimentos.

presentados en este capítulo, la latencia se define como el tiempo que un sistema o un usuario tiene que esperar para recibir el valor de un sensor después de haberlo solicitado. Para obtener las latencias, se utilizó el log del software Android Studio, ya que permite el cálculo de la diferencia de tiempo entre los eventos relacionados con cuando el usuario presiona un botón para obtener el valor actual de un parámetro y cuando dicho valor está listo para que aparezca en la pantalla del teléfono inteligente. Las marcas de tiempo (timestamps) en Android se expresan en mili segundos, que dan la suficiente precisión para los experimentos diseñados.

Para llevar a cabo las mediciones, se desarrolló una aplicación específica para Android. La aplicación envía ciertas etiquetas de texto para identificar fácilmente los eventos dentro del registro (log). Por lo tanto, la medición de la latencia cuando se establece una conexión se señaliza con las etiquetas de "acción UP" (que se envían al pulsar un botón en la aplicación de Android) y "[CHR] - DATA-DIGITAL" (que indica que los datos están listos para ser mostrado en la pantalla). Para los experimentos con el protocolo LP4S-6, la misma etiqueta "ACTION UP" se envía primero, pero la segunda etiqueta es "[TLM] -DATA-ANALOG", que también indica que hay datos de algún sensor listo para ser mostrado. Por lo tanto, la diferencia entre las marcas de tiempo asociadas con la llegada de las dos etiquetas es la latencia estimada. Por ejemplo, la figura 3.17 muestra parte de un log utilizado para obtener la latencia de un beacon. En la misma figura del log se pueden apreciar otras etiquetas relacionadas con tramas Eddystone como son, la URL, UID y frames TLM. Las etiquetas marcadas en los recuadros de color rojo están involucrados en el cálculo de la latencia que en el ejemplo mostrado fue de 0,911 s.

Además de obtener un registro detallado de todos los eventos, Android Studio permite capturar imágenes en tiempo real, tales como los incluidos en las Figuras 3.18 y 3.19, que

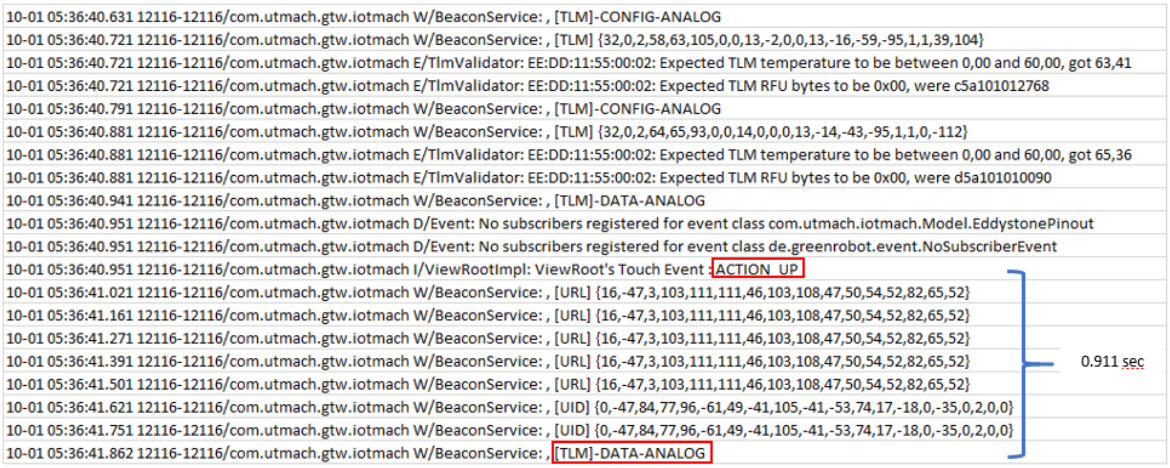


Figura 3.17: Example of Android log for measuring latency.

Figura 3.18: Estableciendo una conexión cambiando de ventanas (izquierda) o usando la misma ventana (derecha).

Figura 3.19: Accediendo a la información embebida a través de la app Android.

ayudarán a aclarar las acciones tomadas en cada experimento. La figura 3.18 muestra las acciones que se realizan cuando se mide la latencia en el caso de establecer una conexión con el mote BLE. En dicha figura, en el caso de uso de la izquierda, el usuario selecciona uno de los motes BLE descubiertos y se abre una nueva pantalla para interactuar con el mote, ya sea para enviar comandos o para leer valores de los sensores. Por lo tanto, la latencia sería el tiempo transcurrido desde que el usuario selecciona un mote en la primera pantalla, hasta que se visualice el valor del sensor en la segunda pantalla. En la tabla 3.5, este escenario es referido como “Conectar con cambio de ventanas”.

En figura 3.18, a la derecha, se ilustra un escenario en el que se muestran los datos del sensor en la misma pantalla después de conectar. Por lo tanto, en este caso, la latencia se mide desde que el usuario presiona el botón para conectar, hasta que el valor del

sensor se actualiza en la pantalla. En tabla 3.5 los resultados asociados a este escenario se hace referencia como Çonectar misma ventana".

En cuanto a las mediciones de latencia para los experimentos basados en beacons, que se realizaron a través de la misma aplicación Android, que también es capaz de interpretar los frames TLM de un Eddsytone con el fin de obtener los valores de los sensores insertados siguiendo el protocolo propuesto. Una vez que el usuario selecciona un beacon descubierto (en la figura 3.19, a la izquierda), se abre una nueva ventana que muestra la información estándar Eddystone de un frame TLM, pero cuando se pulsa el botón "+" (marcado con un círculo) se abre otra ventana (en la figura 3.19, a la derecha) que muestra los valores de los sensores. En este caso, la latencia se mide desde el momento en que el usuario presiona el botón "+", hasta que los valores de los sensores se muestran en la pantalla. Tenga en cuenta que para estas mediciones no se requiere conexión con el beacon, por lo que el usuario sólo tiene que esperar a la llegada de la trama TLM adecuada.

Para la obtención de las latencias, hasta 50 mediciones se promediaron para cada escenario. Es interesante señalar que las latencias obtenidas seguían una distribución normal. Por ejemplo, en la figura 3.20 se puede observar las latencias obtenidos al evaluar un mote que utiliza un intervalo de 100 ms: las latencias se pueden modelar como una curva de campana gaussiana con media igual a 0,834 s y una desviación estándar de 0,511 s. Es importante tener en cuenta que tales curvas pueden utilizarse para modelar el comportamiento de los motes y luego generar muestras artificiales de las distribuciones ajustadas para llevar a cabo simulaciones de Monte Carlo.

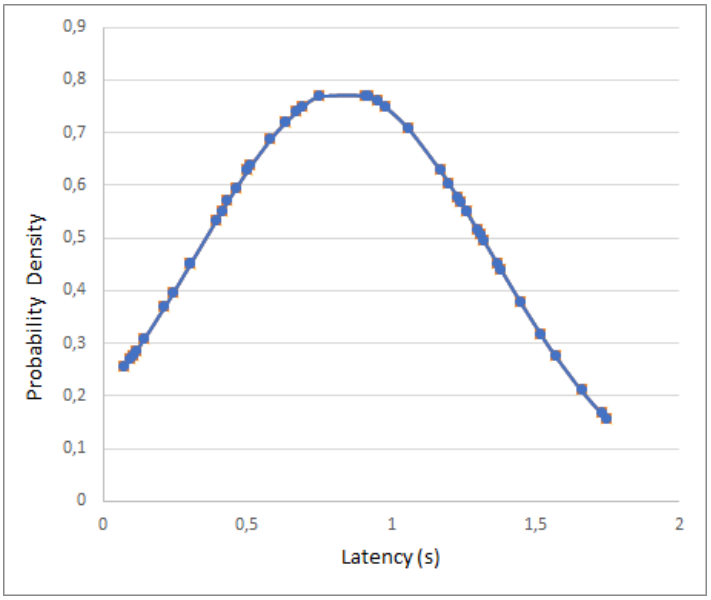


Figura 3.20: Distribución de latencia para un intervalo de 100 ms beaconing interval.

Tabla 3.5: Resultados de latencias para los diferentes experimentos propuestos.

Experimentos		Latencias (s)
Beaconing Interval	100 ms	0.83
	250 ms	1.67
	500 ms	4.66
	1,000 ms	11.7
	2,000 ms	24.21
	5,000 ms	61.18
GATT	Connect Changing Windows	1.05
	Connect Same Window	6.72

La tabla 3.5 muestra las latencias obtenidas para los ocho escenarios considerados. En el caso de las pruebas de beacons, se puede observar que la latencia crece a medida que aumenta el intervalo de balizamiento. Los resultados permiten concluir que los intervalos de 100 ms y 250 ms son útiles para aplicaciones en tiempo casi real o en tiempo real, pero los intervalos más grandes no son adecuados para este tipo de aplicaciones. En cuanto a los experimentos del GATT (es decir, cuando se establece una conexión con el mote), se puede afirmar que la conexión se realiza en el escenario de "cambio de ventanas" es claramente más rápido que para la "misma ventana". Esto se debe al hecho de que el enfoque de "la misma ventana" tiene que descubrir primero y luego conectarse al mote, mientras que en la versión de "Cambio de ventanas", el tiempo transcurrido se mide después que el mote ya ha sido descubierto.

3.5.3. Consumo de potencia del mote

Con el fin de obtener el consumo de energía del mote en cada uno de los ocho escenarios propuestos anteriormente, se utilizó el Nordic PPK. Tal kit consiste primero en una tarjeta electrónica dedicada insertada en el nRF51-DK y luego a un ordenador a través del USB. La aplicación de escritorio PPK de Nordic se ejecuta en el ordenador, lo que permite obtener mediciones precisas y actualizadas del consumo del mote. La interfaz PPK facilita el análisis del consumo de energía mediante las herramientas incluidas para ajustar las ventanas de tiempo, obtener el consumo medio en un intervalo, o para acercar y alejar. Además, los datos capturados pueden ser guardados en un archivo CSV con el fin de realizar luego un análisis más profundo con otras herramientas de software como Matlab.

Las figuras 3.21 y 3.22 son ejemplos significativos de las mediciones de corriente obtenidas. En dichas figuras se puede observar una notable diferencia entre los cuatro casos mostrados. La figura 3.21 compara el consumo de un mote BLE basado en las características del GATT antes y después de la conexión: el cual solo consume 35 A

cuando no está conectado y hasta 15mA cuando se establece la conexión, aunque el consumo medio es en realidad 1.12 mA.

La figura 3.22 muestra el consumo para un mote que utiliza intervalos de 100 ms y 5000 ms. Para un intervalo de 100 ms, hay picos de consumo de hasta 7.5 mA (el promedio es de 865 μA), mientras que con intervalos de 5000ms los picos medidos son similares (de hasta 7 mA), pero el consumo medio desciende a 125 μA.

La tabla 3.6 contiene el consumo de corriente obtenida para diferentes intervalos de transmisión y para los experimentos GATT propuestos. La segunda columna muestra el consumo de corriente media, mientras que el resto de las columnas contiene estimaciones sobre el número de meses que un dispositivo duraría con diferentes tipos de baterías comerciales. Debemos aclarar que este cálculo asume que las baterías se descargan

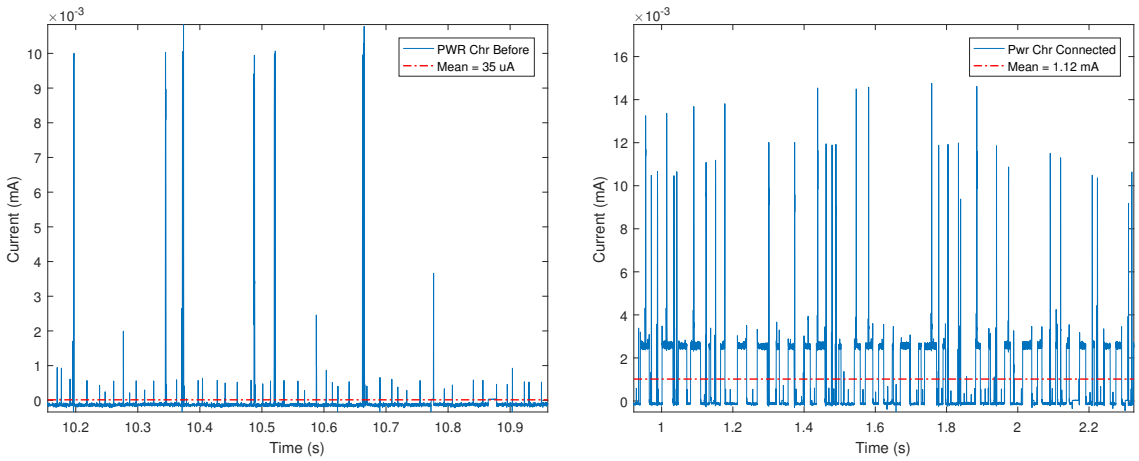


Figura 3.21: Consumo de potencia para un GATT antes de conexión (izquierda) y un GATT conectado (derecha).

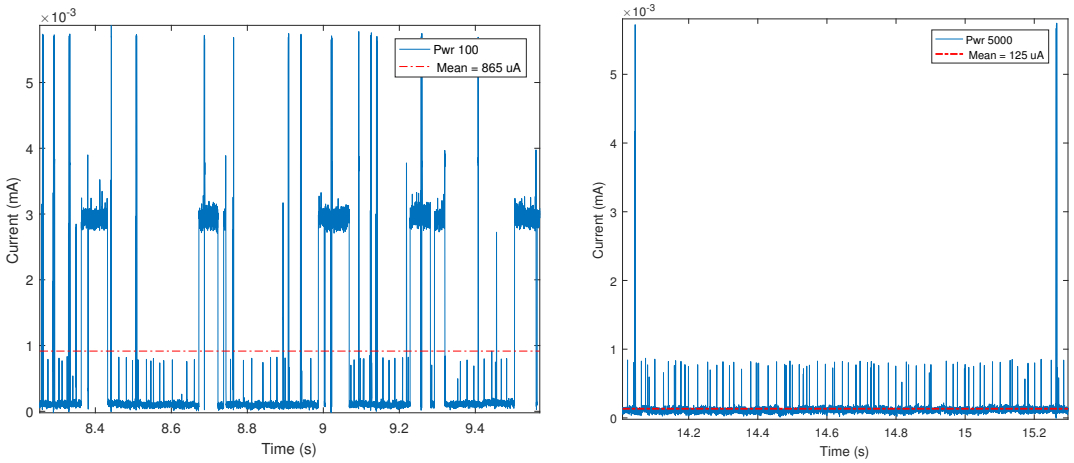


Figura 3.22: Consumo de potencia para intervalos de 100 ms (izquierda) y 5,000 ms (derecha).

Tabla 3.6: Resultados de los consumos de potencia.

Experimentos		Consumo Potencia (μA)	CR2032 240 mAh (meses)	CR2477 1000 mAh (meses)	VNR1582 5000 mAh (meses)
Beaconing Interval	100 ms	865	0.385	1.606	8.028
	250 ms	465	0.717	2.987	14.934
	500 ms	265	1.258	5.241	26.205
	1000 ms	195	1.709	7.123	35.613
	2000 ms	135	2.469	10.288	51.44
	5000 ms	125	2.667	11.111	55.556
GATT	Connect Changing Windows	1115	0.299	1.246	6.228
	Connect Same Window	1115	0.299	1.246	6.228
	Before Connecting	35	9.524	39.683	198.413
	Average	575	0.58	2.415	12.077

completamente, lo que no suele ser el caso, ya que la mayoría de las baterías sólo pueden mantener una tensión adecuada hasta que su capacidad esté entre el 20 % o el 30 % del total. Por lo tanto, la expresión utilizada en la Tabla 3.6 para el cálculo de la duración de la batería es:

$$Battery\_life\,(months) = \frac{Battery\_capacity\,(mAh)}{\frac{Mote\_average\_consumed\_power\,(\mu A)}{1000}} * 24 * 30 \tag{3.1}$$

Con respecto a las mediciones en beacons, se puede observar fácilmente cómo al aumentar el intervalo de trasmisión disminuye significativamente el consumo de corriente. En el mejor de los casos, para un intervalo de 5000 ms, una batería VNR1582 duraría 55 meses (más o menos cuatro años y medio), que es aproximadamente cuatro veces más que los 12 meses que un mote basado en el GATT alcanzaría en promedio con la misma batería.

En el caso de los experimentos del GATT, la corriente en stand-by es tan baja que incluso con la batería más pequeña el dispositivo podría durar varios meses en dicho modo. El consumo de energía es claramente superior cuando se conecta el mote, debido a la potencia consumida por el modulo de comunicación BLE, pero tenga en cuenta que, en general, los motes permanecen la mayor parte del tiempo en stand-by a la espera de conexiones.

3.5.4. Análisis de los resultados

La sub-sección anterior mostró conclusiones pertinentes con respecto a la latencia y el consumo de corriente, pero es interesante conocer a profundidad para una mejor comprensión de la relación entre ambos factores.

En primer lugar, es útil visualizar los resultados de las tablas 3.5 y 3.6 en la forma mostrada en la figura 3.23. En dicha figura las líneas rectas representan el consumo de corriente para los diferentes experimentos GATT, siendo el de la parte inferior el relacionado con el consumo cuando se desconecta el mote, y en la parte superior, la que representa la corriente consumida cuando se establece una conexión. Por lo tanto, esta última línea recta representaría el caso cuando el mote está siempre conectado,

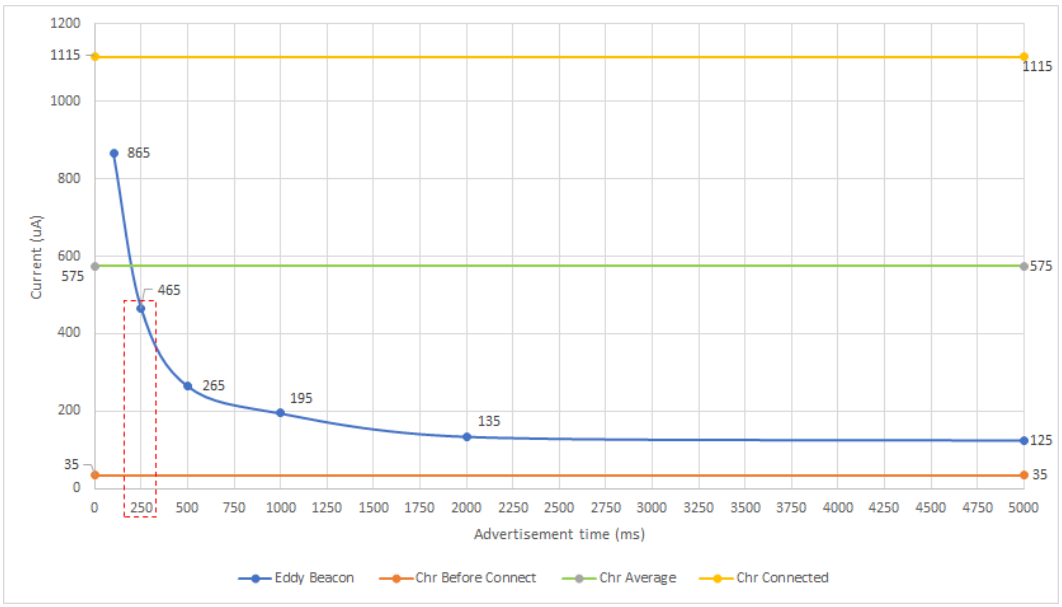


Figura 3.23: Consumo de corriente versus intervalos de transmisión del beacon para los experimentos realizados.

que en la práctica no es eficiente, pero que podría ser el mejor escenario, cuando varios usuarios acceden de forma secuencial al mismo dispositivo. Por el contrario, la línea central es la media de las otras dos líneas rectas y representa una situación más realista que uno o más usuarios se conectan esporádicamente a un mote.

Con respecto a la curva 'Eddy Beacon', que muestra el comportamiento de consumo de corriente de un beacon para los diferentes intervalos de trasmisión ensayadas. Comparando la curva con los tres niveles de referencia a partir de los experimentos del GATT, se puede concluir que la solución de beacons que implementa LP4S-6 obtiene mejores resultados que cuando se utilizan las características del GATT en escenarios reales con uno o varios usuarios que se conectan esporádicamente. De hecho, todos los valores de consumo obtenidos a excepción de la que está asociada a un intervalo de 100 ms, están por debajo de la línea central ('Chr media'). Las ganancias son aún mayores cuando es posible aumentar el intervalo de transmisión del beacon, aunque parece que, con el hardware probado, hay un piso de consumo después de 2000 ms donde corriente consumida no disminuye significativamente.

También es importante darse cuenta de que los protocolos de la familia LP4S no limitan el número de usuarios simultáneos y el consumo del mote no se ve afectado por dichos usuarios, ya que no se requiere una conexión con el dispositivo para obtener los valores de los sensores y, por lo tanto, todos los usuarios reciben en paralelo la transmisión de datos. Este hecho significa que, cuando varios usuarios acceden a un mote sin necesidad de una conexión, la corriente sería cerca de la línea recta ('Chr

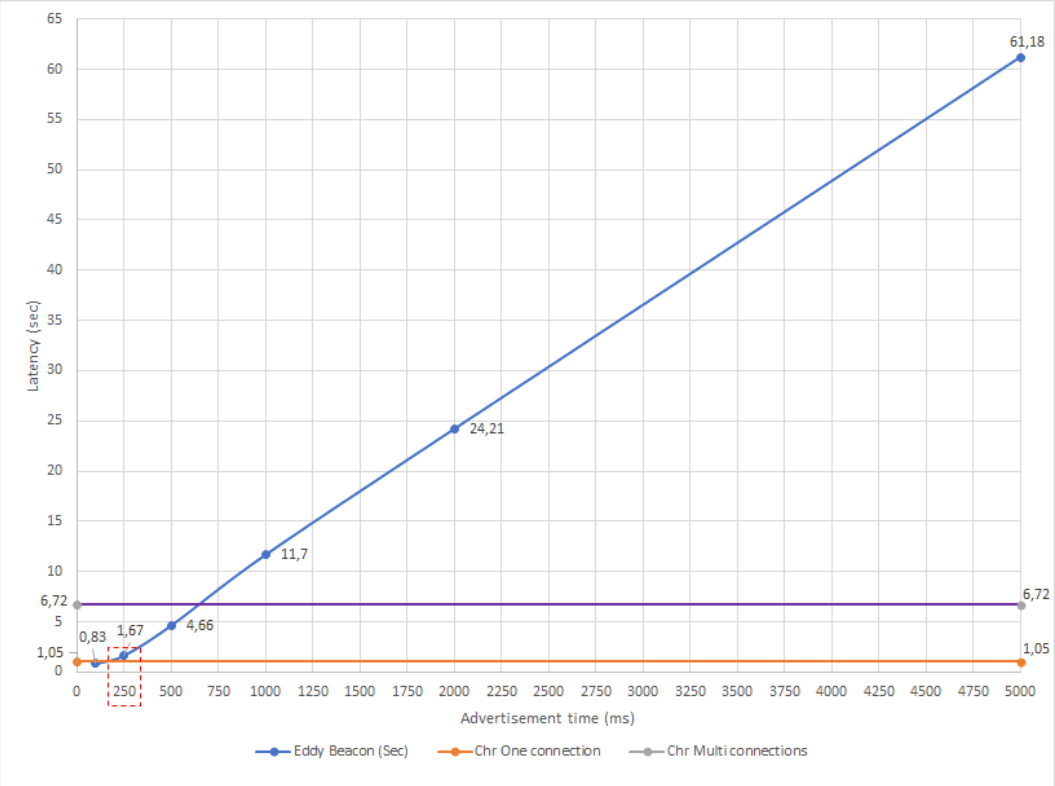


Figura 3.24: Latencia versus intervalos de tiempo (advertisement).

before Connect') que se muestra en la figura 3.23 (1,115 μA), lo que implica que incluso el intervalo de 100 ms (865 μA) sería más eficiente en términos de energía.

Con respecto a la latencia, la figura 3.24 permite la comparación de los resultados para motes basados en GATT y basados en beacons en relación con el intervalo de transmisión. Una vez más, las líneas rectas sirven como referencia y representan las latencias mostradas en la tabla 3.5 para los experimentos GATT, mientras que la curva 'Eddy Beacon (Sec)' contiene los datos para el enfoque propuesto basado en LP4S-6. Así, la línea 'Chr One connection' representa el mejor escenario logrado cuando sólo un usuario está conectado al mote, mientras que la línea 'Chr Multiconnections' representa la latencia cuando varios usuarios se conectan al mismo mote. En figura 3.24 se puede también observar que el beacon LP4S-6 muestra durante 100 ms y 250 ms una latencia mejor que el escenario GATT, siendo aún menor en el caso de 100 ms. Sin embargo, para los intervalos de 1000 ms, 2000 ms y 5000 ms, se supera claramente los 6,72 s de latencia obtenido para el escenario GATT multi-usuario.

Por último, cabe destacar que en las figuras 3.23 y 3.24 hay un rectángulo de puntos que indica la solución que ofrece el mejor equilibrio entre la latencia y el consumo de energía. Sin embargo, es importante hacer hincapié en que la selección del protocolo de comunicaciones del mote depende altamente de la aplicación. Por ejemplo, en una

aplicación de telemetría sin restricciones de potencia, donde sólo un usuario concurrente está conectado al mote y que requiere velocidades de muestreo bajas, se recomienda tanto la aproximación GATT y los enfoques con intervalos de 100 ms y 250 ms. Sin embargo, en la misma situación, si hay varios usuarios, el GATT no sería una buena opción. No obstante, en general, para aplicaciones de control y telemetría a distancia, el uso del GATT para establecer una conexión suele ser la mejor opción.

Un escenario diferente surge cuando los motes son alimentados con baterías e instalados en lugares remotos donde el mantenimiento regular es difícil o caro. En tal escenario, los enfoques basados en beacons tienden a ser la mejor alternativa. Por otra parte, los parámetros a se sensados determinan el intervalo de transmisión y la batería puede ser elegida siguiendo los resultados obtenidos en las tablas 3.5 y 3.6, y teniendo en cuenta las dimensiones y peso máximo del mote. Un ejemplo del escenario descrito puede suceder en una aplicación de agricultura de precisión, donde un intervalo de 5000ms sería suficiente para medir factores como la temperatura ambiente o l humedad del suelo, que en la mayoría de las situaciones cambian lentamente. Además, en este tipo de escenarios, por lo general no hay restricciones de tamaño, por lo que una batería de gran tamaño podría ser embebida dentro del mote.

3.6. Conclusiones

En este capítulo se presentó la familia de protocolos LP4S ligeros, que se centran en proporcionar comunicaciones de baja latencia y bajo consumo de energía. Además, LP4S permite la auto-detección y auto-registro del mote a través de mecanismos de plug-and-play para sistemas de telemetría IoT. El uso de motes BLE para telemetría implica dos tipos de conectividad: pueden ser conectados directamente a un servidor GATT a través de características o se pueden utilizar beacons con la ayuda del protocolo LP4S. Para determinar qué tipo de conectividad es la mejor opción para ser utilizado en diferentes contextos y aplicaciones, varios experimentos se llevaron a cabo.

Se demostró la viabilidad de utilizar el protocolo LP4S-6, embebido en un beacon comercial Eddystone. Los resultados demostraron la superioridad de la solución propuesta en términos de consumo de energía y latencia cuando varios usuarios están conectados a un mote. Por otra parte, los resultados también sugirieron el uso del protocolo LP4S-6 para escenarios en los que la latencia no era una restricción, pero el bajo consumo de energía era esencial.

Finalmente, es importante mencionar que, aunque la familia protocolo fue desarrollada y probada dentro de un protocolo restrictivo como los beacons Eddystone, también puede implementarse en otros dispositivos IoT similares, proporcionando comunicaciones eficientes y rápidas.

Capítulo 4

Arquitectura plug and play basada en TEDS virtuales para la Web de las cosas (PnP-VTED)

4.1. Introducción

Actualmente, vivimos en un mundo conectado al Internet, donde el Internet de las cosas (IoT) dejó de ser un término novedoso para convertirse en lo cotidiano. A través de la infraestructura que IoT nos ofrece, con dispositivos físicos o virtuales perfectamente identificables e inter operando en una red de datos incluso sin la interacción humana [195], múltiples dominios de aplicación y negocios emergentes se han desarrollado rápidamente. Ejemplo de ellos son las Smart cities, Smart Home, e-Health, Wearables, Power and environment, Agricultura de Precisión, Industria 4.0, entre otras. Las aplicaciones en estos dominios son cada vez más disímiles exigiendo mínimos tiempos de respuestas, seguridad y calidad de servicio.

Dados los requerimientos de estos dominios de aplicación de una mayor abstracción de las cosas, el IoT ha tenido que evolucionar y los sensores inteligentes además ser capaces de procesar la información de los sensores in chip, realizar cierta lógica de cómputo, intercambiar información a través de redes alambricas o inalambricas, incorporar actuadores para accionar motores o mostrar información recibida o procesada en un display. Además, estos deben incorporar tecnologías plug and play que permitan reconocer a los sensores activos.

En los último años, los sensores inteligentes han reducido su tamaño, por lo que también son llamados motes, y a pesar del avance actual del harvesting [196], siguen dependientes de una batería para su correcto funcionamiento, siendo importante conseguir alargar su vida útil para la autonomía de todo el sistema. Por tanto, se hace necesario

optimizar el firmware del sensor implementando sistemas operativos en tiempo real, Real Time Operating System (RTOS), tecnologías de comunicación y protocolos ligeros que permitan un bajo consumo, baja latencia y Quality of Service (QoS).

En el IoT market, los desarrolladores de tecnología se encuentran en una verdadera jungla a la hora de elegir un proveedor, donde cientos de fabricantes generan prototipos, soluciones y SDK propietarios, que hacen casi imposible lograr integrar sensores inteligentes de diferentes fabricantes en una aplicación o sistema IoT. Por otro lado, tenemos el aumento progresivo de la comunidad Do it yourself (DIY), con la necesidad de crear sus propios productos y personalizar desarrollos de IoT. Los cuales también son un motor de impulso al IoT, pero teniendo que luchar a diario con un IoT Landscape muy diverso, donde se carece de aplicaciones y estándares de representatividad de los motes, lo cual hace difícil la interoperabilidad en todos los niveles, desde la parte física hasta las aplicaciones.

Dando respuesta a esta necesidad aparece la Web of Things como concepto que hace una abstraction layer above the IoT, rehusando los ya existentes protocolos web que han permitido la escalabilidad e interactividad de las aplicaciones, permitiendo a los desarrolladores web hacer uso de los datos y servicios que brindan los objetos en IoT. Mientras que IoT resuelve problemas de networking, Web of Things (WoT) está enfocada exclusivamente en los protocolos y herramientas a nivel de aplicación [197].

A pesar de todos estos antecedentes y tendencias aún existen las siguientes problemáticas a resolver:

- Carencia de soluciones con mecanismos plug and play a nivel de nodo sensor.
- Necesidad del uso de estándares y semántica para la descripción de los motes que permitan la Interoperabilidad entre sensores de diferentes fabricantes y tecnologías de comunicación y entre aplicaciones de un mismo dominio o entre dominios.
- Necesidad de protocolos ligeros a nivel de los motes que permitan mecanismos de detección y autorregistros para la comunicación motes - gateways, gateway - cloud y motes - cloud.

Este trabajo da una solución a dichas problemáticas con tres aportes importantes:

- Se presenta una arquitectura basada en el estándar IEEE 21451 que propone diferentes modificaciones relacionadas con el concepto de Virtual Transducer Electronic Data Sheet (VTEDS). El sistema propuesto permite mecanismos plug-and-play (Plug-And-Play (PnP)) en la capa de sensores de un ecosistema IoT.

- También se presenta una aplicación web con una interfaz gráfica intuitiva que permite monitorear, controlar y administrar todos los sensores y la arquitectura de comunicaciones.
- El sistema propuesto se evalúa empíricamente en diversos escenarios reales. Los experimentos realizados utilizan diferentes nodos de sensor para mostrar que la arquitectura diseñada es muy rápida al desplegar sensores e intercambiar datos.

Este capítulo está basado en la publicación [198] y organizado de la siguiente forma. Sección 4.2 hace una revisión del estado del arte y trabajos relacionados con los estándares de representatividad de los motes y plug and play. En la sección 4.3 se presenta un visión general de la arquitectura PnP-VTED propuesta. Sección 4.6 muestra implementación de la arquitectura. Sección 4.7 describe la configuración experimental y analiza los resultados de los experimentos realizados. Finalmente, la Sección 4.8 está dedicada a las conclusiones.

4.2. Estado del arte y trabajos relacionados

En esta sección se revisan los trabajos relacionados de la comunidad científica que nos permitan determinar los requerimientos básicos esperados para dispositivos IoT y cuales han sido los protocolos ligeros definidos para dispositivos IoT.

4.2.1. Estándares para interoperabilidad de redes de sensores

El desarrollo de sistemas IoT para lograr entornos informáticos ubicuos deben estar caracterizado por requerimientos de heterogeneidad, escalabilidad, interoperabilidad, flexibilidad, fiabilidad y disponibilidad. Para afrontar estos retos, sobre todo los de heterogeneidad e interconectividad de los sensores, es necesario que estos sean diseñados en función de estándares ampliamente aceptados por la industria [199]. En este contexto los principales referentes son: el estándar IEEE 21451, definido por los gigantes International Organization for Standardization (ISO), Network Information Service (NIS), Institution of Electrical and Electronic Incorporated Engineers (IEEE) y el estándar Sensor Web Enablement (SWE) definido por el Open Geospatial Consortium (OGC).

4.2.1.1. Estándar ISO/IEC/IEEE 21451

El estándar IEEE 1451 fue creado por la IEEE Instrumentation and Measurement Society's TC-9 Technical Committee on Sensor Technology, que luego pasó a llamarse IEEE 21451, con el objetivo de estandarizar la conectividad en los sensores inteligente a diferentes instrumentos y sistemas a través de una red local o Internet facilitando la

interoperabilidad de dispositivos y datos [200]. Esta es una de las principales iniciativas de implementación de sensores plug-and-play.

El estándar establece tres componentes básicos de un transductor inteligente (smart transducer): Transducer Interface Module (TIM), Network Capable Application Processor (NCAP) y Transducer Electronic Data Sheet (TEDS). La figura 4.1 muestra la estructura básica de un smart transducer basado en el IEEE 1451.

TIM (Transducer Interface Module)

El TIM es donde se ubican los sensores y actuadores y actúa como una interfaz con la instrumentación real y con los procesos de automatización. Este módulo también procesa la señal obtenida de los sensores antes de enviarla al NCAP.

Un TIM puede aceptar tanto a un smart transducer (tiene TEDS embebidos) o transducer convencionales que son la gran mayoría del mercado y a pesar de la heterogeneidad de fabricantes y estándares de hoy en día, estos pueden ser agrupados y clasificados en 6 grupos, los cuales son las interfaces: analógica, digital, Universal Asynchronous Receiver-Transmitter (UART), 1-Wire, Serial Peripheral Interface (SPI) y Inter-Integrated Circuit (I2C) [201].

NCAP (Network Capable Application Processor)

Es el dispositivo de red encargado de comunicar al smart transducer con un nivel superior donde se encuentran la cloud computing y las aplicaciones de usuarios, es decir, puede actuar como un gateway de comunicación de datos.

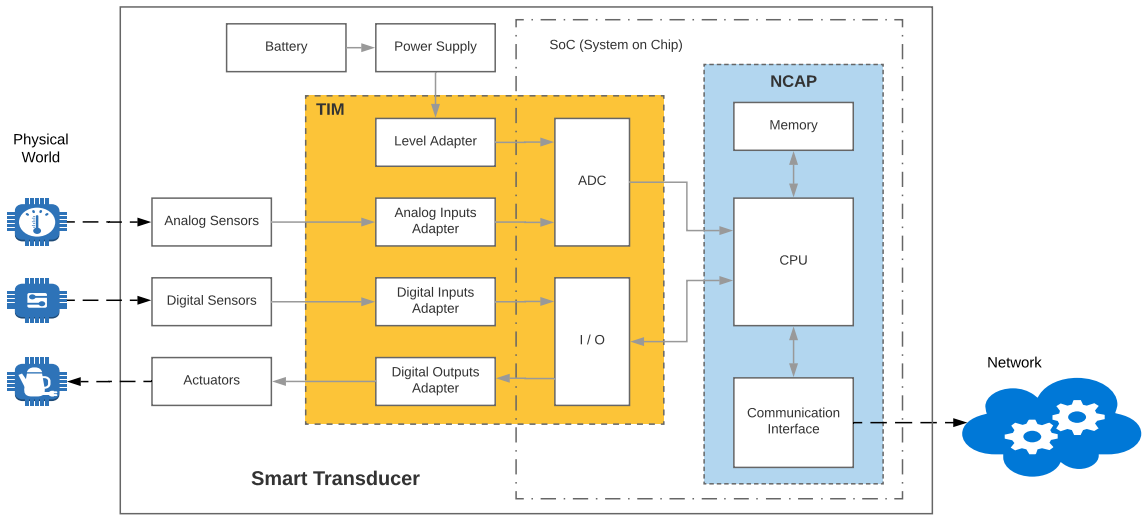


Figura 4.1: Estructura básica de un Smart Transducer.

En [201] se enfatizan las características que debe tener un NCAP inteligente para conseguir un mecanismo plug and play (P&P), donde el NCAP automáticamente puede detectar y calibrar los transductores conectados, permitiendo la interoperabilidad y adaptabilidad. Estas características del NCAP son:

- Conexiones físicas y eléctricas estándares entre el NCAP y los transductores
- Identificación del transductor para ser detectado por el NCAP
- Diagnóstico del transductor por el NCAP

TEDS (Transducer Electronic Data Sheets)

Los TEDS son los encargados de describir a los transducer a través de un conjunto de metadatos que se asemejan a las hojas de datos (datasheets) que brindan los fabricantes de componentes electrónicos. Ejemplos de estos metadatos son ID del dispositivo, nombre del fabricante, número serial y datos de calibración.

Los TEDS en su inicio estaban embebidos (en memoria) en los transductores (smart transducers) y garantizaba los principios básicos plug and play, permitiendo la capacidad de auto-identificación (self-identification) y auto-descripción (self-description) de los transductores. No obstante esta solución tenía ciertas limitaciones, ya que cada transductor tenía su propio estándar, requiriendo de un hardware complejo en el TIM y NCAP, lo cual elevaba el costo para implementar los TED, aumentaba la complejidad y el costo de la solución. Además, el aumento del número de transductores aumenta directamente la complejidad y el costo. En esta solución añadir, eliminar o modificar un transductor, requiere de un proceso largo y la necesidad de un experto, lo cual aumenta nuevamente la complejidad y costos.

Como solución a esta problemática aparecieron los virtual TEDS, donde las tablas de los sensores pueden estar en otros dispositivos de la red, sea en el NCAP o en un servidor dedicado. Esta solución tiene también sus limitantes, dado el requerimiento de una biblioteca TED y dado que los fabricantes no publican los TEDS de sus transducers. La labor de actualización y calibración en tiempo real de los mismos es un reto a la hora de implementar este tipo de solución. Nuestra propuesta se basa en el uso de los Virtual TEDS como veremos más adelante.

El estándar ofrece una lista de plantillas (templates) ya definidos que cubren la mayoría de los casos, no obstante es posible definir los propios. En [202] podemos ver la estructura de dichos templates y en la tabla 4.1 podemos observar algunos ejemplos de estas plantillas.

Tabla 4.1: Ejemplos de TED Templates.

Tipología	Nombre	Valor ID
Basic TEDs	Manufacturer ID	43 (Acme Company)
	Model number	7115
	Version letter	B
	Version number	1
	Serial number	X0 1891
Standard templates	Accelerometer y Force	25
	Charge Amplifier (w/ attached accelerometer)	26
	Charge Amplifier (w/ attached force transducer)	43
	Microphone with built-in pre amplifier	27
	Microphone pre amplifier (w/ attached microphone)	28
	Microphones (capacitive)	29
	High-Level Voltage output Sensors	30
	Current Loop Output Sensors	31
	Resistance Sensors	32
	Bridge Sensors	33
	(LVDT/RVDT) Sensors	34
	Strain Gauge	35
	Thermocouple	36
	Resistance Temperature Detectors (RTDs)	37
	Thermistor	38
	Potenciometric Voltage Divider	39
Calibration template [9]	Calibration table	40
	Calibration Curve (polynomial)	41
	Frequency Response Table	42
User Data [9]	User name	John Smith
	User ID	123456
	Location	Machala
	Other information	Project 1.2

Interfaces de comunicación

IEEE 21451 define un set de estándares para las interfaces del TIM y NCAP. Cada uno de los elementos de esta familia constituye un estándar en sí, los estándares son: IEEE21451.0 que define funcionalidades comunes, Comandos y TEDS, IEEE21451.1 define servicios de Network, IEEE21451.2 define serial interfaces (UART, USB, I2C, SPI), IEEE 21451.3 define los web services, IEEE21451.4 : define MMI (mixed-mode interface), IEEE21451.5 : define interfaces para protocolos inalámbricos, como son IEEE 802.11x (WiFi), IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (ZigBee), and 6LowPAN. El protocolo 6LowPAN es un (IPv6 compatible) que permite acceso directo de transductores desde Internet, IEEE21451.7 define interfaces para Radio Frequency Identification (RFID)

y IEEE21451.1.4 : define interfaces para eXtensible Messaging and Presence Protocol (XMPP). La figura 4.2 muestra un resumen de esta familia.

En aras de incorporar mecanismos P&P, un unificado web service para IEEE 1451, un smart transducers [203–205] fue desarrollado para acceder a la información de los transductores a través de la web. Para ello debe existir un cliente web services en el lado del nodo sensor y un servidor web services. La implementación del web services añadió un nuevo reto en la forma de representar la información de los transductores para ser enviadas por la web. La primera solución con datos codificados fue el Extensible Markup Language (XML) ofrecida por la World Wide Web Consortium (W3C) [206] que añadió alto tráfico y complejidad de la solución por la verbosidad del formato de la cabecera HTTP. Para resolver esto apareció Binary XML by Open Geospatial Consortium (OGC), este formato puede ser también usado para comprimir los datos de sensores y ha sido definido usando OGC's y GML (geography markup language) [207]. Otra iniciativa fue la Fast Infoset por el International Organization for Standardization (ISO) y la International Telecommunication Union (ITU), definiendo los estándares ITU-T (X.891) y ISO (IEC 24824-1) [208], los cuales para lograr un bajo ancho de banda requieren un alto grado de compresión de la data para lograr un alto rendimiento entre la conexión web service del NCAP con el server. Como solución a la anterior problemática W3C regresa con un Efficient XML interchange (EXI), que incorpora un algoritmo relativamente simple, que es susceptible de implementación rápida y compacta, y un pequeño conjunto de representaciones de tipo de datos [209].

Como se puede ver el uso de REST con web services y XML, tiene un costo alto de complejidad e incorporan una alta tasa de transferencia de datos (throughput). Nuestra propuesta usa diferentes protocolos de comunicación y una adaptación de los Virtual TEDS que permiten implementar una solución plug and play con baja latencia, bajo throughput, garantizando calidad de servicio como veremos más adelante.

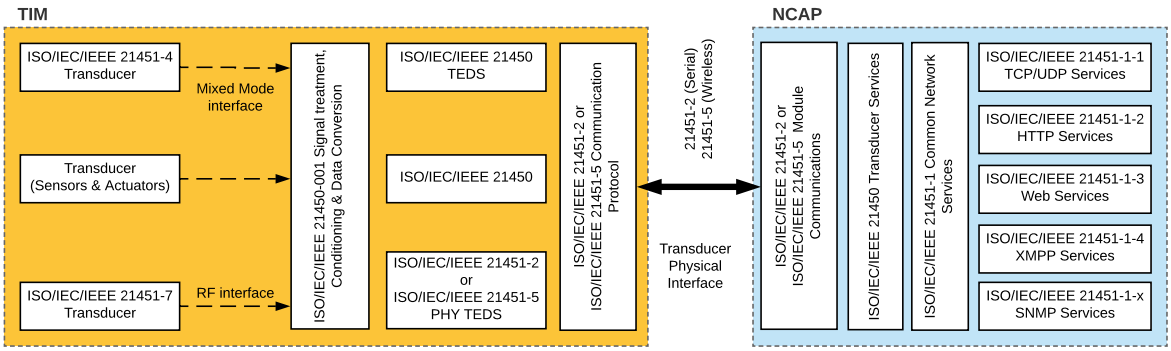


Figura 4.2: Familia del estándar IEEE 21451

Trabajos relacionados con el IEEE 21451 standard

Algunas implementaciones del estándar han sido llevadas a cabo desde la pasada década hasta la actualidad, como es el caso de [210] que implementa un sistema con sensores plug and play con auto-calibración compatible con el estándar IEEE 1451.4. Así como, [211] muestra el desarrollo e implementación de un sensor analógico de temperatura (tipo resistivo) basado en IEEE 1451-4 (Mixed Mode Interface). En este caso el sensor de temperatura embebía los TEDS en una memoria DS2433. El NCAP fue implementado en dos módulos separados, un microprocesador de 8 bits y una aplicación en LabView de National Instruments corriendo en una PC. Por otro lado, [212] implementa una red Zigbee para el control de iluminación basada en el IEEE 21451, donde los TIM son los end node con las luminarias y los coordinadores requeridos en la red Zigbee son conectados vía USB a un PC formando ambos el NCAP del estándar. La arquitectura propuesta usa REST web services y un programa en MATLAB que corre en la PC para decodificar la metadata recibida. Otro ejemplo de uso del estándar con Zigbee lo encontramos en [213], donde añaden también un GPS, utilizan la misma estrategia de usar como NCAP la unión de un coordinador más PC corriendo en este último una aplicación de LabView para decodificar la metadata de los TEDS.

En [214] se implementa un solución IoT haciendo uso de Virtual TEDS y comandos del IEEE 21451-4 para la interpretación de ModBus devices, mientras que [215] demuestra el uso del estándar IEEE 21451-001 en el tratamiento de la señal de smart transducers, donde incorpora además un algoritmo de segmentación propio Real-Time Segmentation and Labeling (RTSAL) para el procesamiento de la señal y una tabla de algoritmos denominada Transducer Algorithm Data Sheet (TADS). Dicho algoritmo fue simulado en MATLAB y a pesar de su recomendación en microprocesadores denota un cierto requerimiento de recursos para ser implementados en motes reales.

Otro ejemplo de implementación del estándar IEEE 21451 en un sistema para monitorear la calidad del aire utilizando módulos de comunicación inalámbrica GSM fue encontrado en [216] y en [217] se desarrolla un programa en Python para generar una plataforma independiente de TEDS basados en el IEEE 1451-0. Aquí, también hacen referencias a trabajos previos desarrollados en C, C++, C#, Delphi y LabView. El programa tiene una estructura modelo vista - controlador y la representación de los TEDS fue realizada tanto en formato binario como XML. En este caso utilizaron un Arduino como TIM y una Raspberry Pi como NCAP.

Una iniciativa interesante de incorporar la semántica a wearables implementados con el IEEE 21451 standard nos brinda [218]. La ontología allí propuesta (TEDSOnto) está basada en Protege and SPARQL, brindando inter-operatividad a nivel semántico. A pesar de que esta solución solo fue simulada, demuestra claramente una tendencia

de representación de los smart sensor a nivel semántico. Otro ejemplo que define una ontología para IEEE 1451 para describir a los sensores y establecer un link entre las especificaciones IEEE 1451 y las operaciones de un Open Geospatial Consortium Sensor Observation Services (OGC SOSs) aparece en [219], mientras que en [220] se implementa un sistema de Home Automation basado en el uso de Transductores Inteligentes que usa mecanismos plug and play basados en el estándar IEEE 21451 para la representación de los smart sensors en su sistema. A través de una API (Get - Request y JSON) se puede acceder al gateway (ITE) donde hay un websocket server y obtener de ahí los datos de los sensores registrados. Como hemos visto en otras referencias el uso de técnicas Get-Request introducen latencias al sistema. En nuestro trabajo usamos técnicas Pub-sub para contrarrestar estas problemáticas.

4.2.2. Estándares OGC

OGC reúne a varias organizaciones públicas y privadas, responsables de definir estándares abiertos y de interoperabilidad, principalmente para Sistemas de Información Geográfica (SIG) y la web. Las siguientes subsecciones analizan algunas de las normas de OGC relacionadas con las aplicaciones de telemetría que permiten la descripción del sensor y la interoperabilidad.

4.2.2.1. OGC-Sensor Web Enablement Framework

La otra gran alternativa que también ha sido estandarizada y utilizada por la comunidad científica para lograr la inter-operatividad es la SWE (Sensor Web Enablement), desarrollada y mantenida por Open Geospatial Consortium [221], bajo la premisa que todos los sensores reportan posición, todos conectados a la web, todos con metadatos registrados, todos leíbels remotamente, algunos controlados remotamente y habilitando los estándares HTML y HTTP para el intercambio de información en la web [222], [223].

El framework SWE consiste en una arquitectura formada por un set de estándares que definen el formato del dato de forma similar a una interfaz web services [224]. Entre los estándares encontramos los siguientes: Observations & Measurements Schema (O&M): define un esquema para codificar la data de los sensores, Sensor Model Language (SensorML) y Transducer Markup Language (TransducerML or TML): permiten describir los sensores, Sensor Observations Service (SOS) y Sensor Planning Service (SPS): permiten el enlace entre el usuario y las mediciones de los sensores, Sensor Alert Service (SAS): publica y escucha (publishing and subscribing) las alertas de los sensores y Web Notification Services (WNS): permite enviar mensajes o alertas desde los SAS y SPS hacia los usuarios. La tabla 4.2 muestra una clasificación según [224] de los estándares OGC antes mencionados.

OGC mantiene además los programas OGC Interoperability Program (IP) y OGC Specification Program y su OGC Network una infraestructura accesible vía Internet que implementa los estándares OGC.

La figura 4.3 muestra una arquitectura OGC-SWE, en la que podemos identificar perfectamente el rol de cada uno de los estándares OGC-SWE y su interacción, así como las aplicaciones, capas de sensores, que habilitan a los sensores para que estos sean accesibles y controlables a través de Internet.

XML es una parte clave de esta infraestructura y todos los servicios y modelos de contenido se especifican utilizando esquemas XML [222]. SensorML proporciona una rica colección de metadatos que se pueden usar para descubrir los sistemas de sensores y las características del proceso de observación. Estos metadatos incluyen identificadores, clasificadores, restricciones (tiempo, legal y seguridad), capacidades, características, contactos y referencias, además de entradas, salidas, parámetros y sistemas de localización [201].

Tabla 4.2: Estándares OGC.

SWE Information Model	SWE Service Model
SWE Common	Sensor Observation Service (SOS)
Sensor Model Language (SensorML)	Sensor Alert Service (SAS)
Observations and Measurements (O&M)	Sensor Planning Service (SPS)
Transducer Markup Language (TML)	Web Notification Service (WNS)

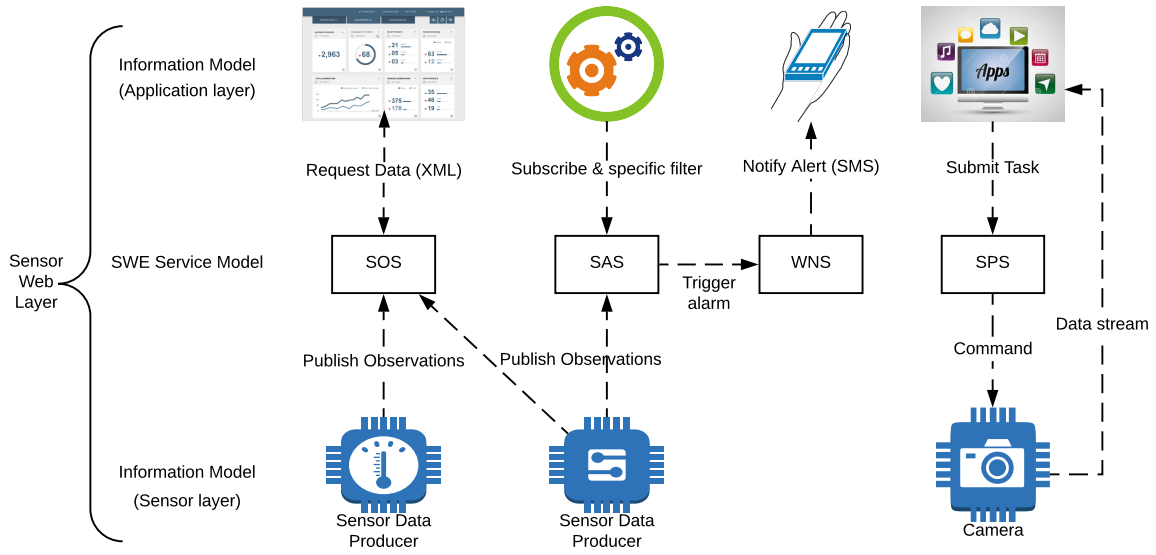


Figura 4.3: Arquitectura OGC-SWE

Trabajos relacionados con SWE

Las especificaciones SWE de OGC han sido adoptadas por la Administración Nacional de Aeronáutica y del Espacio (NASA) en su sistema de redes geo-sensoras de observación de la Tierra (EO-1) [205]. SWE también se ha implementado de manera efectiva sobre IEEE 1451 NCAP, lo que ha resultado en el más alto nivel de interoperabilidad y funcionalidad inteligente compatible tanto con sensores en nodos de sensores como con nodos de sensores dentro de una red [201], [222].

En [225], describen como el uso de una notación semántica de registro de Sensores y las solicitudes de inserción de observaciones pueden servir como base para la verificación de la coherencia basada en el razonamiento y, por lo tanto, mejorar el proceso de coincidencia manual, que permiten descubrir dispositivos y datos en sistemas basados en SWE. Mientras en [226] se implementa un Ubiquitous Sensor Network (USN) para la integración de carros inteligentes incluyendo en su propuesta varios servicios de OGC-SWE. El autor plantea problemáticas en los tiempos de respuesta cuando se usan grandes repositorios y se requieren reacciones rápidas.

La meteorología ha sido un nicho para SWE, un ejemplo de ello es [227], que implementa una red de sensores hidro-meteorológicos basados en SWE. El modelo envía la información espacio temporal de los sensores en un esquema O&M para describir una observación y sus medidas, SensorML para describir un sensor, SWE Common para enviar datos comunes de los servicios SWE y Transducer Markup Language (TML) para enviar datos temporales de las medidas. Como podemos ver la implementación de estos sistemas es compleja y requiere de un equipo de expertos para hacerlas funcionales. El autor menciona los conflictos actuales en la automatización del ingreso de las observaciones al servidor y recomienda añadir ontologías del tipo Semantic Sensor Network (SSN) [228]. Otro ejemplo de uso del SWE modelo para implementar aplicaciones IOT relacionadas con señales topológicas, geoespaciales, etc. la encontramos en [229]. Donde podemos encontrar conceptos como WSW, Worldwide Sensor Web entre otros. En este trabajo se evidencia una de las falencias de esta solución y es en función de las latencias de la arquitectura REST (request / response) empleada en SWE. En nuestro diseño en cambio hacemos uso de la arquitectura Publish / Subscribe (MQTT) como alternativa de solución. Como hemos podido observar SWE permite describir la arquitectura de los motes a expensas de una alta complejidad de implementación.

4.2.2.2. OGC-SensorML

Sensor Model Language (SensorML) [230] es un modelo general y un esquema de codificación XML para sensores y procesos de observación. SensorML, además de ser parte del framework OGC SWE, también se puede usar de forma independiente. La

función principal de SensorML es la definición de procesos de observación basados en las mediciones recopiladas. Para tal fin, proporciona descripciones de los sensores, actuadores y filtros tales como su geoposición, los valores observados (datos medidos) o su precisión, entre otros. También define una cadena de proceso ejecutable para derivar observaciones.

En SensorML, todos los componentes se modelan como procesos. Los sensores y actuadores se consideran componentes del proceso y las plataformas se modelan como sistemas. Todos estos componentes pueden participar en cadenas de procesos, que son procesos con entradas, salidas y parámetros.

Hay dos grandes grupos de procesos: procesos físicos, que están formados por detectores, actuadores y sistemas de sensores, y procesos no físicos o "puros", que pueden manejarse como operaciones matemáticas.

Los modelos en SensorML están codificados como esquemas XML. Por lo tanto, los documentos de instancia XML se crean para todos los componentes y procesos de observación. La figura 4.4 muestra un ejemplo de XML adaptado de [230] donde se define una latencia de 10 s para un sensor específico.

```
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML/1.0"
    xmlns:swe="http://www.opengis.net/swe/1.0"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/sensorML/1.0
    http://schemas.opengis.net/sensorML/1.0.0/sensorML.xsd" version="1.0">
    <member xlink:role="urn:ogc:def:role:OGC:detector">

    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!-- Latency in Air Value           -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <parameter name="latency" xlink:arcrole="urn:ogc:def:property:OGC:latencyTime">
        <ConditionalValue>
            <condition name="medium">
                <swe:Category definition="urn:xogc:def:property:OGC:medium">
                    <swe:value>air</swe:value>
                </swe:Category>
            </condition>
            <data>
                <swe:Quantity definition="urn:ogc:def:property:OGC:duration">
                    <swe:uom code="s"/>
                    <swe:value>10</swe:value>
                </swe:Quantity>
            </data>
        </ConditionalValue>
    </parameter>
```

Figura 4.4: Ejemplo de XML para definir el valor de latencia de un sensor.

4.2.2.3. Protocolo OGC-PUCK

El estándar OGC Programmable Under Water Connector (OGC-PUCK) [231] define un protocolo que se utilizará en instrumentos con puertos de comunicación RS-232 y Ethernet. Al igual que SensorML, el protocolo PUCK pertenece a la suite OGC-SWE y se usa principalmente para almacenar y recuperar la descripción de un instrumento. No obstante, también se puede usar independientemente de los otros estándares SWE. Una característica interesante es que PUCK puede incluir los TEDS IEEE 21451 descritos en la Sección 4.2.1.1 como descriptores. Por lo tanto, PUCK puede utilizar documentos descriptivos en formato OGC-SWE SensorML o IEEE 21451 TEDS como un código çontrolador"para un instrumento.

Un cliente que puede interpretar el protocolo PUCK puede recuperar automáticamente los metadatos del instrumento. Normalmente, tanto el código SensorML como el controlador del instrumento serán almacenados en la memoria PUCK del instrumento antes de la implementación. Cuando el instrumento está encendido, el host recupera los metadatos para identificar el instrumento. Tal proceso de identificación se llama plug-and-work y se describe en la siguiente subsección.

4.2.2.4. OGC-Plug-and-Work mechanism

La figura 4.5 resume el flujo y el conjunto de elementos y estándares OGC implicados en el mecanismo de plug-and-work de OGC [232], que es el proceso para detectar e identificar un instrumento para integrarlo automáticamente en un sistema de observación cuando el instrumento está encendido. Entonces, un cliente web puede recuperar los datos de observación a través de Internet.

Este mecanismo comienza a nivel del instrumento con el modelado del instrumento a través del Sensor Interface Descriptor (SID), que se basa en el estándar OGC SensorML, a través del cual se describen los protocolos y el formato de datos de cada instrumento. Entonces, estas descripciones (archivos SID) se almacenan en la memoria del instrumento en sí mismo a través del Puck payload, que incluye tanto TEDS y archivos SID (archivos XML), como se puede observar en la figura 4.5. Luego, en el nivel de host (Sistema de observación), se implementan módulos para que se pueda detectar un instrumento habilitado para PUCK. El módulo detector PUCK se utiliza en el nivel RS-232, que puede detectar la velocidad en baudios de la comunicación serie RS-232, mientras que el módulo ZeroConfig permite detectar instrumentos basados en IP. Una vez que se detecta un instrumento a través del intérprete SID, todos los metadatos de los sensores se extraen y se analizan desde sus archivos XML. Por lo tanto, el intérprete SID actúa como una puerta de enlace. Finalmente, los datos de observación recuperados están disponibles en el nivel web, a través del conjunto de servicios web (SAS, WNS, SPS)

disponibles a través del estándar SWE. Por lo tanto, la combinación de PUCK con SID habilita la capacidad plug-and-work. Se pueden encontrar más detalles sobre los estándares OGC y el mecanismo OGC Plug-and-Work en [233].

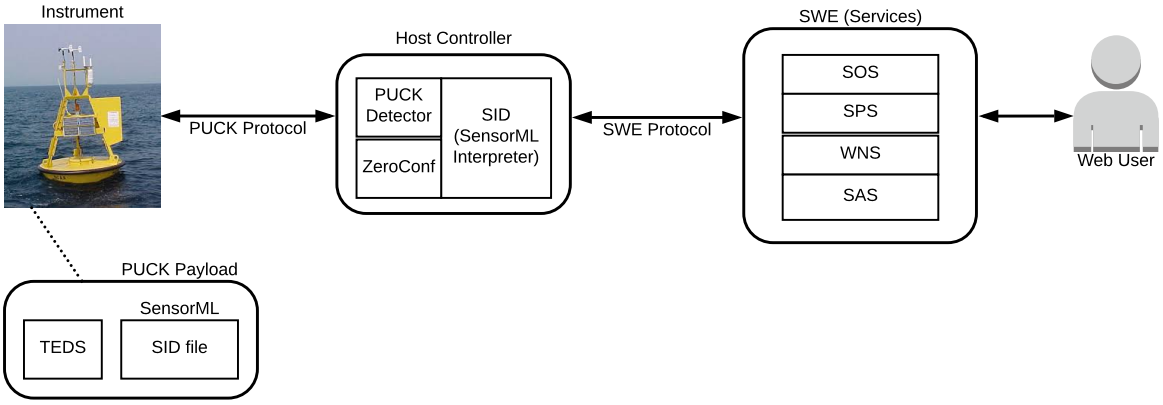


Figura 4.5: Arquitectura Plug-and-work usando OGC-PUCK y SID.

4.2.3. Otras alternativas de solución Plug and Play e inter-operatividad

Existen otras iniciativas que no hacen uso de los dos estándares mencionados pero que han aportado al desarrollo del IoT añadiendo mecanismos que permitan la inter-operatividad en IoT. Un ejemplo de Plug and play nos brinda [234], donde lo que se identifican son módulos de hardware y no a los transductores en sí. Aquí, cada módulo tiene un descriptor del hardware, lo cual implica que si se añaden nuevos sensores al nodo, entonces hay que modificar offline dichos descriptores para que el software del main board pueda describirlos. Otra solución que implementa plug and play a través de un hardware que actúa como un hub de periféricos que denomina μPnP-Mesh networking aparece en [235], donde los periféricos son identificados utilizando una red de componentes eléctricos pasivos (resistores). Una vez detectados los módulos conectados físicamente al hub, la aplicación desarrollada sobre Contiki, solicita a un gateway los drivers necesarios para la identificación de los motes conectados a las redes de los periféricos conectados. Los drivers fueron generados bajo un lenguaje propietario, implementando complejos procesos. La detección de primaria de periféricos por hardware es una limitante apreciable de esta solución, además de la complejidad en su implementación.

En [236] se propone un modelo ligero para la anotación semántica de datos utilizando dispositivos heterogéneos en IoT para proporcionar anotaciones para datos en aplicaciones de salud. Para conseguir una interoperatividad semántica hacen uso de Resource Description Framework (RDF) [237] para la representación de los datos y SPARQL

para leerlos. El sistema fue simulado con Tableau, Gruff-6.2.0 y Mysql. Este artículo demuestra una tendencia marcada en el uso de una web semántica para lograr la deseada interoperatividad en IoT.

Mientras que [238] introduce un modelo de arquitectura con cinco patrones para lograr la interoperatividad de IoT a gran escala, donde intervienen proveedores de plataformas, de cosas, usuarios, desarrolladores, etc. El proyecto [239] aunque está en estado piloto es una importante iniciativa en lograr la ansiada abstracción de las plataformas de hardware, dominios de aplicación y servicios. Ejemplos de aplicaciones IoT y arquitecturas semánticas encontramos en [240] y [241]. Por ultimo, [242] es un excelente trabajo que nos muestra la evolución y retos que ha tenido la interoperatividad desde la IoT, pasando por la WoT y terminando en la Semantic Web of Thing (SWoT) [243]. Un caso de estudio de la SWoT la encontramos en el proyecto europeo FIESTA-IoT [244], donde implementan un motor semántico para la notificación de datos WoT y un generador SWoT en el contexto de OpenIoT [245].

4.2.4. Análisis de los trabajos relacionados

Después de revisar los trabajos relacionados, es posible llevar a cabo un análisis comparativo de la solución propuesta en este documento y los estándares mencionados anteriormente, enfatizando las ventajas de la nueva propuesta.

En primer lugar, es importante tener en cuenta que tanto el estándar IEEE 21451 como el OGC ofrecen opciones que garantizan la interoperabilidad del sensor, incluida la implementación de mecanismos plug-and-play como los descritos anteriormente. Tales estándares hacen uso de conceptos y recursos similares, siendo posible combinarlos. Por ejemplo, TEDS puede usarse tanto con IEEE 21451 como con soluciones basadas en OGC para describir los metadatos de los sensores. De forma similar, la solución presentada en este documento, que se basa en el estándar IEEE 21451, utiliza el mismo metadescriptor (TEDS) que el estándar OGC-PUCK. Otra similitud radica en el uso de un protocolo para encapsular los descriptores: en este documento se utiliza el protocolo ligero para sensor (LP4S), que es similar a OGC-SensorML.

No obstante, las implementaciones de OGC-PUCK e IEEE 21451 a menudo almacenan el descriptor en el sensor o instrumento, lo que puede plantear un problema al actualizar la información. Esto se resuelve a través de VTEDS, que agrega la posibilidad de usar repositorios para archivos SID externos. Los archivos SID utilizados como meta-descriptores deben crearse antes de la implementación de la solución. La creación de archivos SID sin una herramienta de soporte puede ser un proceso tedioso y propenso a errores. Un ejemplo de una aplicación visual SID Creator se encuentra en [246]. Por el contrario, en el sistema VTEDS propuesto, todos los parámetros del sistema se crean

a través de una interfaz web a la que se accede a través de Internet, lo que permite su uso inmediato. Cada vez que VTEDS cambia, el sistema genera publicaciones en el tópico correspondiente, de tal forma que el nodo sensor consume las actualizaciones realizadas en tiempo real, a través de los mecanismos que se explicarán más adelante en la sección 4.3.3. Una ventaja adicional del enfoque presentado es que se pueden reutilizar los descriptores disponibles en la nube entre diferentes aplicaciones de dominio cuando sea necesario, actualizando y agregando nuevos VTEDS de forma transparente para el nivel del sensor.

En cuanto a la solución OGC-PUCK, tiene dos limitaciones importantes cuando se trata de la implementación de proyectos modernos de telemetría basados en nodos de sensores inalámbricos con restricción de potencia. La primera limitación es la cantidad de memoria y los recursos de procesamiento necesarios para implementar los mecanismos plug-and-play cuando el PUCK payload está integrado en el nodo del sensor. A veces, el PUCK payload debe guardarse en una tarjeta SD como en [247]. Además, OGC-PUCK no está optimizado para ser utilizado en dispositivos de restricción de potencia con poca memoria y procesamiento. Por ejemplo, OGC-PUCK requiere incluir controladores para administrar puertos RS-232 o Ethernet. Este problema es abordado por el sistema propuesto con el uso del protocolo LP4S, que puede integrarse incluso en dispositivos de hardware con restricciones de recursos como beacons Eddystone (se puede encontrar más información sobre el protocolo LP4S en [124]).

La segunda desventaja es que hay demoras significativas en la detección de hardware cuando se obtiene mediciones. Por ejemplo, en el caso de PUCK-RS232, el software PUCK realiza un proceso cíclico para sincronizar las velocidades del puerto serie del instrumento con el host, lo que deriva en latencias adicionales. Un proceso similar ocurre en el módulo Zeroconfig para instrumentos IP-PUCK. Se produce otra demora cuando el mecanismo utilizado para obtener los datos de observación recupera los datos del sensor de un archivo SID. Ambos procesos están bien explicados en [233]. En comparación, la solución propuesta basada en VTEDS resuelve dicha limitación de manera rápida y eficiente, como se muestra en la sección 4.7.

4.3. Arquitectura Plug and play basada en Virtual TEDS

En esta sección se presenta la arquitectura general del sistema propuesto, que permite describir a los nodos sensores con mecanismos plug and play. Como se detallará en las subsecciones siguientes, se eligió el estándar IEEE 21451 para describir el hardware del sensor con una adaptación sobre el uso de VTEDS.

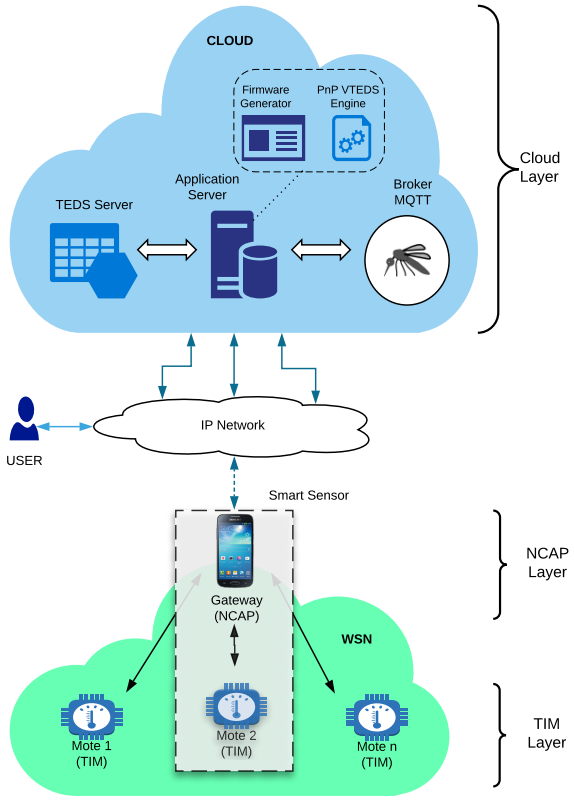


Figura 4.6: Esquema general del sistema plug and play basado en VTEDS.

4.3.1. Visión global

En la figura 4.6 se muestra dicha arquitectura, donde podemos identificar claramente los elementos propios del estándar, dígase los TIM y NCAP. En este caso los motes o nodos sensores constituyen los TIMs del sistema, conectados en una especie de intranet de sensores denominada WSN, por la naturaleza de la comunicación inalámbrica de los sensores. Los TIMs encargados de monitorear las variables del proceso en que serán usados, por ejemplo, temperatura o humedad envían los valores leídos hacia la cloud a través de NCAP, formado por un gateway IoT, que permite interpretar la información encapsulada en el protocolo del TIM para luego encapsularla y enviarla a la cloud según el protocolo establecido por las aplicaciones y servicios residentes en la cloud. El NCAP, en la mayoría de los casos, salvo cuando el mote tenga dispositivos de red incorporados, habilita la información en una red IP, para ello muchas veces es necesario enlazar diferentes protocolos y tecnologías de comunicación.

El servidor que aloja los VTEDS también están en la nube, donde también hay aplicaciones IoT responsables de generar el firmware para los nodos del sensor y los servicios necesarios para descubrir automáticamente los TIM activos. La base de datos de

VTEDS alojada en la nube está disponible a través de servicios web para las aplicaciones de monitoreo del sistema y para cualquier otra aplicación de IoT que lo requiera.

4.3.2. Sistema de mensajería y protocolos de comunicación

En los últimos años, se han propuesto diferentes protocolos de mensajería para interconectar nodos de IoT heterogéneos. Tradicionalmente, el problema de la heterogeneidad se resolvía implementando servicios de intercambio de archivos, implementando bases de datos compartidas o utilizando mecanismos Remote Procedure Call (RPC), pero los últimos desarrollos tenían como objetivo la creación de sistemas de mensajería.

Los sistemas de mensajería centralizan las comunicaciones para minimizar el acoplamiento entre los componentes de los sistemas distribuidos, como los que suelen implementarse en los desarrollos de IoT. Este desacoplamiento y el uso de comunicaciones asíncronas permiten llevar a cabo comunicaciones más robustas: los componentes que deben comunicarse no tienen que estar funcionando al mismo tiempo, por lo que delegan la entrega de los datos a un sistema de mensajería, lo que les permite concentrarse sobre qué información enviar en lugar de cómo enviarla.

La mayoría de los sistemas de mensajería utilizan dos modelos para conectar un transmisor y un receptor: el modelo de cola peer-to-peer o el modelo de publicación/-suscripción (pub/sub). El modelo de cola peer-to-peer permite que el transmisor envíe un mensaje a una cola donde un receptor en particular lo leerá. En el caso del modelo de publicación/suscripción, los mensajes se envían a uno o más destinatarios que han expresado previamente interés en ellos. En la práctica, cuando el transmisor envía (publica) un mensaje, en realidad se envía una copia a un receptor (suscriptor). Ejemplos de sistemas de mensajerías populares (pub/sub) implementan Java Messaging Service (JMS) (p. Ej., OpenJMS, WebSphereMQ), Advanced Message Queuing (AMQP) (p. Ej., Apache Qpid, SwiftMQ, RabbitMQ), Extensible Messaging and Presence (XMPP) (p. Ej., Ejabberd, Apache Vysper) o Message Queuing Telemetry Transport (MQTT) (p. ej., ActiveMQ, Mosquitto).

MQTT [70] es especialmente interesante para IoT, ya que se ha diseñado como un protocolo de mensajería abierto que permite intercambiar mensajes con dispositivos de restricción de recursos (es decir, sensores, actuadores, teléfonos móviles) que operan en redes de alta latencia. Debido a estas razones, MQTT es muy popular entre los desarrolladores de IoT, que lo han usado en aplicaciones de smartwatch [248], robótica [249], domótica [250] o healthcare [251].

Debido a las ventajas mencionadas anteriormente, en el sistema propuesto en este capítulo, en la capa de nube, todos los mensajes que provienen de la WSN son administrados por un servidor MQTT (intermediario) que recibe asincrónicamente los datos de

los nodos del sensor remoto. Dicha mensajería, tanto del lado de los TIM como de los NCAP hacen uso de la familia de protocolos Lightweight Protocol for Sensors ((LP4S-6, LP4S-X and LP4-J), que están destinadas a proporcionar comunicaciones livianas a dispositivos de IoT con batería y recursos limitados. Estos protocolos fueron descritos en la sección 3.3 y en [124] y sus características principales incluyen:

- Se pueden incrustar en beacons BLE.
- Pueden enviar datos recogidos de sensores de beacons y recibir comandos para actuadores.
- La familia de protocolos LP4S requiere poca memoria y uso de CPU, lo que le permite implementarse en dispositivos de bajo recurso. Los protocolos también reducen el tiempo de comunicación, disminuyendo así el consumo de energía. Por ejemplo, los protocolos LP4S se probaron con beacons basados en GATT, produciendo un consumo de energía de solo $35\,\mu A$ en modo de espera (antes de conectarse a otros dispositivos) y un promedio de $575\,\mu A$ cuando uno o más usuarios conectados ocasionalmente al mote. En el caso del protocolo LP4S-6, se probó en un beacon Eddystone con intervalos de balizamiento de 100 ms a 5 s obteniendo un consumo de energía de entre 125 y $865\,\mu A$, lo que sugiere que tal protocolo puede usarse en escenarios donde el consumo de bajo consumo de energía es esencial.

4.3.3. Principio de funcionamiento y flujo de datos del mecanismo plug and play

En ésta sección se describen los procesos necesarios que deben ocurrir tanto en los TIM, NCAP y Cloud para que se puedan crear descriptores dinámicos de los sensores y actuadores conectados al sistema IoT, a partir de la información que envían los sensores inteligentes, que permiten descubrir el hardware instalado, cantidad de sensores y actuadores conectados y su naturaleza.

4.3.3.1. Mecanismo plug and play a nivel NCAP

El primer mecanismo plug and play de nuestro sistema comienza en el nivel más bajo, es decir en los TIM, donde se debe primeramente reconocer el hardware instalado, dígase cantidad de sensores analógicos y su ubicación específica en el hardware del mote. De la misma forma se debe reconocer la cantidad de entradas y salidas digitales, ya que en estas últimas estarán conectados principalmente los actuadores del sistema. Este proceso debe ser dinámico y flexible, de forma tal que un usuario pueda configurar un

TIM con los sensores y actuadores propios del dominio de aplicación donde será usado. Por tanto, el firmware del TIM debe ser creado bajo este principio. En la figura 4.7 se muestran los mecanismos de auto-configuración del TIM y su posterior auto-registro en el nivel superior de la arquitectura, es decir el NCAP (gateway) de la WSN. El término auto-registro significa que el propio TIM a través de los mecanismos que explicaremos más adelante es capaz por sí mismo de informar a los niveles superiores los sensores, actuadores y demás parámetros del mismo, para que de forma remota un usuario en la web pueda gestionarlo.

El proceso de descubrimiento del hardware en el TIM se consigue ya que su firmware fue desarrollado de tal forma que a partir de unos descriptores (etiquetas) se define la estructura del TIM, por ejemplo, la ubicación de cada pin, si este debe ser configurado como entrada o salida, digital o analógico, que módulos de comunicación usará, si debe conectarse a otros equipos en su red, se definen las IP, puertos, tiempos de demoras, etc. Estas etiquetas pueden ser modificadas en cualquier momento sin alterar el núcleo del

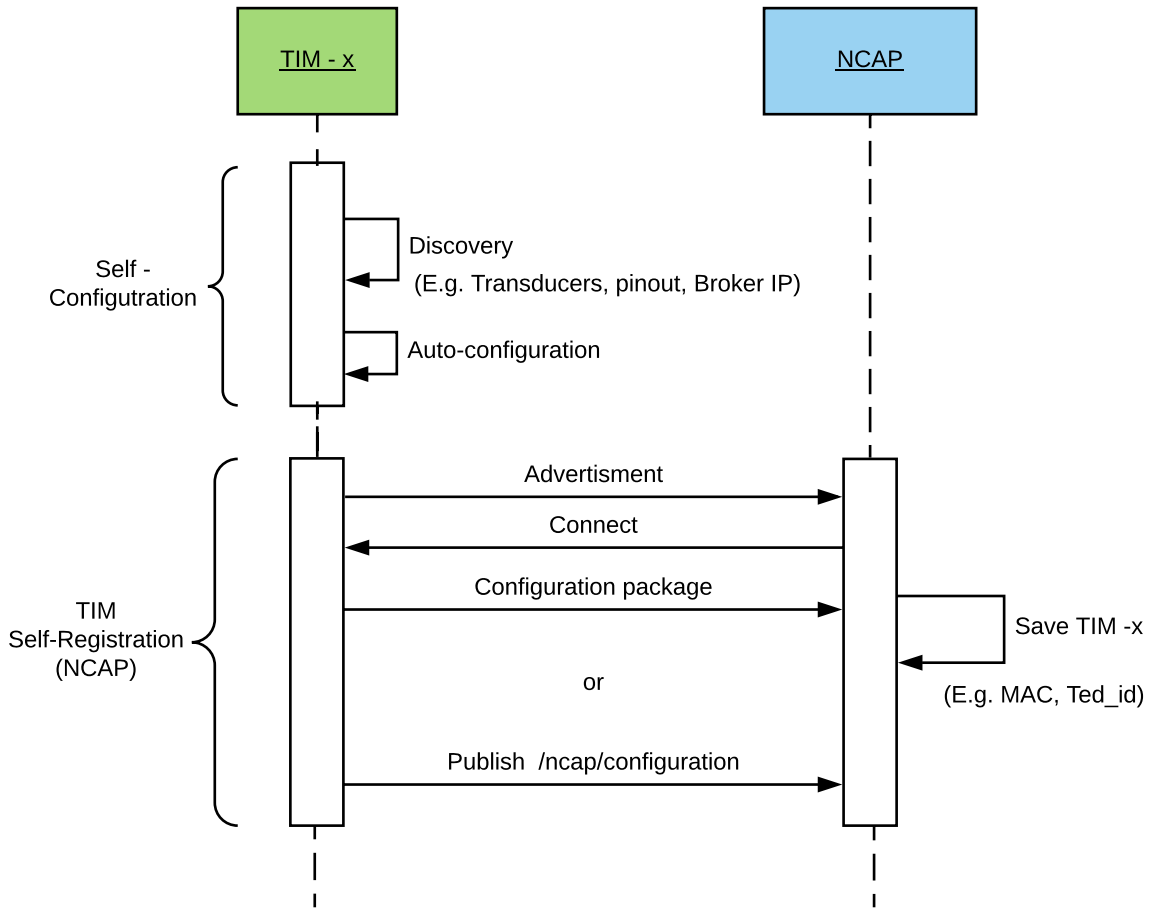


Figura 4.7: Mecanismos de Auto-configuración y auto-registro del TIM a nivel NCAP

firmware. De esta forma, un usuario sin experiencia en programación podría configurar un TIM, solamente dando valor a cada una de estas etiquetas. Por tanto, podemos decir que el firmware del TIM al leer dichas etiquetas ha descubierto los parámetros de funcionamiento y se autoconfigura. Este proceso ocurre cada vez que el TIM se reinicia.

Una vez que el TIM se autoconfigura y está listo para el funcionamiento definido en la etiquetas, es necesario informar hacia el nivel superior (NCAP y Cloud) dicha configuración, para que a esos niveles el usuario pueda acceder al TIM, ya sea para monitorear los valores de los sensores como para enviar un comando hacia los actuadores. O sea, el TIM debe quedar registrado en el sistema. Aquí nuevamente, el proceso es dinámico y transparente para el usuario final, ya que el TIM es capaz de auto registrarse en el sistema, haciendo uso del protocolo LP4S, presente en todas la capas del sistema. El TIM envía una trama de configuración de forma automática y tanto el NCAP como la cloud son capaces de extraer del protocolo todos los parámetros del TIM, para ser guardados en sus bases de datos, lo cual agilizará procesos posteriores (monitoreo o control) sobre dicho TIM. Este proceso se denomina auto-registro.

El FOG level viene dado por la capacidad y nivel de procesamiento que tienen los NCAP actuales, por ejemplo, un smartphone corriendo una aplicación móvil con toda la potencia de procesamiento en sistemas operativos Android o iOS o en computadoras al estilo de los Raspberry-Pi con sistemas operativos Raspbian que permiten aplicaciones en Python o Node-JS, dándole al NCAP suficiente capacidad para gestionar a los TIM. Por lo que, a nivel local un usuario puede monitorear los sensores o actuar sobre el sistema a través de los TIM descubiertos.

4.3.3.2. Mecanismo plug and play a nivel de Cloud

El diagrama de secuencias de la figura 4.8 ilustra el proceso de auto-registro y auto-calibración de un nodo sensor en la cloud. El proceso comienza cuando el NCAP usando el protocolo LP4S-J (descrito previamente en el capítulo 3), enví un JSON a la cloud indicando los parámetros de configuración del TIM. En realidad, el NCAP publica la información del TIM en el broker MQTT de la cloud, donde los servicios (Pnp VTEDS Engine) y las aplicaciones del sistema están suscritos al mismo tópico de configuración. Por lo tanto, los metadatos del TIM se extraen del JSON y se almacenan en la base de datos del servidor. Este proceso se repite para cada uno de los TIM asociados con cada NCAP.

En cuanto al proceso de auto-calibración, éste permite calibrar cada sensor registrado en el sistema de manera dinámica y transparente para el usuario. Por lo tanto, se evita la necesidad de enviar un técnico a la ubicación del nodo del sensor. Esto es especialmente útil cuando los nodos están instalados en lugares remotos. Los parámetros

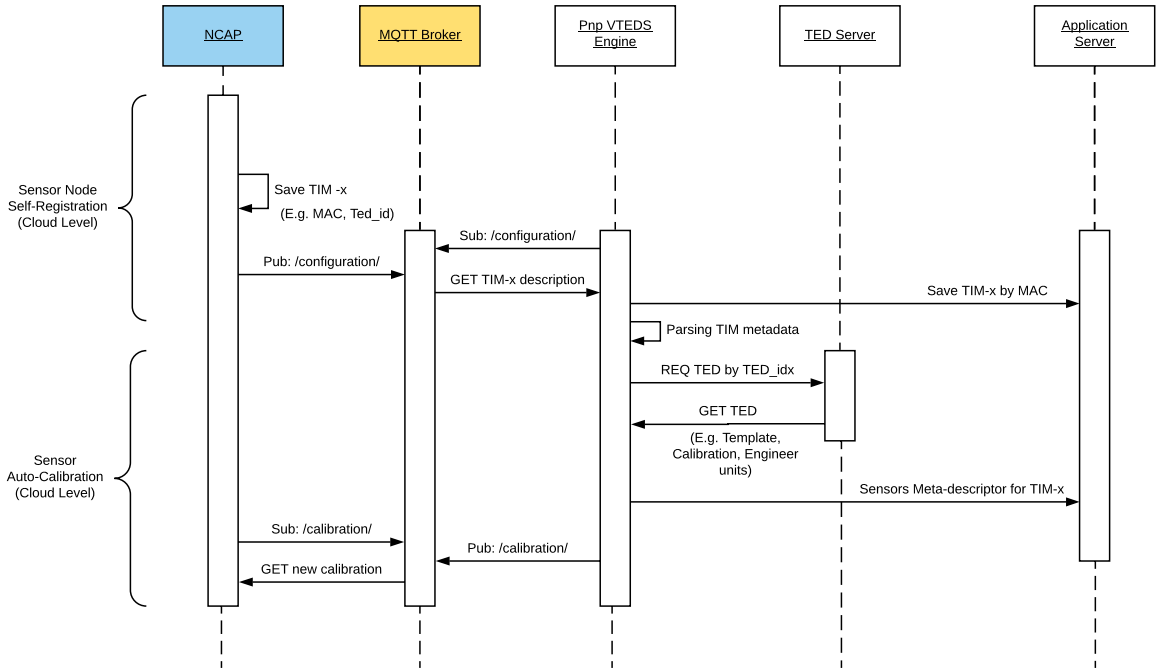


Figura 4.8: Mecanismos de Auto-configuración y auto-registro del TIM a nivel de la cloud.

para la calibración del sensor se ingresan en el sistema a través de un navegador web. La aplicación web guarda todos los valores de calibración en la base de datos, por lo que es posible restaurar en el futuro valores de calibración antiguos. Los nuevos valores de calibración también son administrados por un servicio del sistema, de modo que, una vez que el usuario solicite cambiar la configuración de calibración, el sistema publica tal solicitud en la cola MQTT del tópico de calibración asociado con la MAC del TIM a calibrar. Además, el NCAP está suscrito a los temas de calibración de los TIM descubiertos.

4.4. Metadata de un nodo sensor

En la sección anterior mencionamos nuestra elección por el IEEE 21451 para describir a los nodos sensores (TIM and NCAP) con una adaptación del uso de TEDs virtuales. En está sección indicaremos el proceso de creación de la metadata de un nodo sensor con todos sus descriptores, partiendo del estándar mencionado y con nuestro aporte en los virtual TEDS para aumentar la potencialidad del mecanismo descriptor, dando apertura a nuevas implementaciones descriptoras, incluso semánticas.

En la figura 4.9 se muestra la estructura de la metadata de un nodo sensor que reside en la base de datos del sistema, extraída del protocolo LP4S-J (Json) que envía el NCAP a través del mecanismo descrito en el epígrafe anterior. Es importante destacar,

que el lector puede hacer uso de este protocolo o utilizar el suyo propio para utilizar los mecanismos aquí descritos, dando flexibilidad en la implementación de este sistema. La metadata creada por los servicios residentes en la cloud se puede apreciar en el extremo derecho de la figura, bajo el nombre de Sensor data of Mote 1 y la misma cumple con la estructura propuesta por el estándar IEEE 21451 detallada en el estado del arte de este capítulo. Es decir, Basic TEDS, Standard template, datos actualizados del sensor y otros campos que hemos añadido para potenciar este descriptor. Los nuevos campos entre otros son, dominios de aplicación, calibración, unidades de medida, etc.

Toda esta información es extraída del Json, en el ejemplo de la figura 4.9, dentro del Json de configuración del mote podemos encontrar los siguientes campos correspondientes a los Basic TEDS, nombre que propietario le puso al nodo sensor, MAC única del dispositivo, Identificador de la empresa propietaria del nodo sensor. Además, aparecen dos descriptores correspondientes a los TEDS, y es aquí donde hemos modificado la concepción inicial de los standards template, que fue creada para ser quemadas en el hardware del sensor. Está restricción y limitantes obvias, fueron eliminadas con el uso de virtual TEDS, es decir, estos TEDS al encontrarse en un servidor web, permite una mayor gestión y mantenimiento. en nuestro caso, hemos concebido este parámetro en nuestro diseño con mucha flexibilidad, permitiendo no solo describir a los sensores y actuadores, sino nuevos campos como los dominios de aplicación, muy en boga en el ámbito del Internet de las cosas. Ya que por ejemplo un mismo sensor de temperatura, en dependencia del dominio de aplicación, el valor de temperatura medido puede

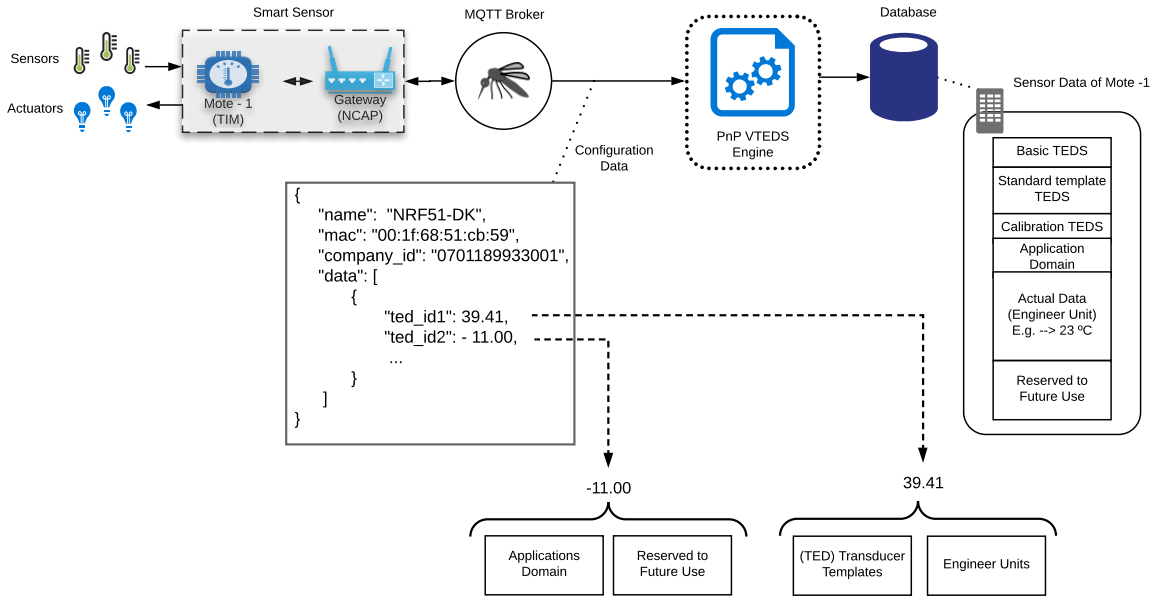


Figura 4.9: Estrucutura de la Metadata de un Smart sensor.

ser utilizado incluso por máquinas de inferencia e inteligencia artificial para toma de decisiones. En caso de un dominio de aplicación de e-health con la temperatura obtenida podría inferirse un cambio de estado del paciente o si el dominio de aplicación es agricultura de precisión entonces con la misma variable podría decidirse abrir o cerrar una electroválvula de riego.

Por tanto, el campo TED, dentro de nuestro diseño toma mayor relevancia y queda abierto para incluir más descripción del nodo sensor, dígase descripciones semánticas entre otras. En el caso específico del ejemplo de la figura solo hemos utilizado dos virtual TEDS, el primero para describir el standard template junto con la unidad de medida y el otro par el dominio de aplicación, pero siguiendo este mismo esquema no hay límites para nuevos descriptores. El primer TED-id con valor de 39.41, describe a un sensor con standard template tipo 39, Potenciometric Voltage Divider y el 41 representa el índice de la tabla de unidad de medida en la base de datos, en la cual puede encontrarse por ejemplo la unidad de medida de Voltaje. El segundo TED.id con valor de -11.00, el campo -11 puede estar representando por ejemplo el dominio de aplicación de Agricultura de Precisión. El signo negativo se incluyó a modo de ejemplo para mostrar la flexibilidad en la descripción y creatividad de nuevos descriptores.

Es importante destacar que el campo TED-id es representado con un número fraccionario con formato de punto fijo 8.8. Esto está dado por el uso del protocolo LP4S diseñado y optimizado para arquitecturas restringidas en cuanto a memoria, CPU y demás recursos, por ejemplo, beacons BLE. Ejemplos de uso de dicho protocolo con beacons fueron mostrados en el capítulo anterior, constituyendo éste una continuación y complemento del anterior dentro del diseño de la arquitectura propuesta.

Como hemos podido apreciar, los campos dentro del JSON son dinámicos de acuerdo a los sensores y actuadores conectados al TIM, donde pueden existir tantos campos como sean necesarios y con varios campos Ted-id para describir mejor al transducer. El caso crítico del Ted-id representado por un número real, es cuando se usa representación fixed 8.8, donde habría 1 byte con signo para los templates, equivalente a 128 templates lo cual es un número grande de templates inclusive. Para unidades de ingeniería sería el byte sin signo o sea 256 diferentes posibles unidades de ingeniería, que es igual una cantidad apreciable. Lo mismo serian 128 diferentes dominios de aplicación. En todos los casos a pesar de la restricción hay un margen en la actualidad para describir casi todos los transductores existentes y comerciales hoy día. Cuando no existe la restricción de cantidad de bytes para representar el número real, entonces, la cantidad de posibilidades se multiplican por cada byte usado es decir por 256, dando la posibilidad de una representación amplia de los transductores. La única limitante estaría en el mote que sea capaz de representar números reales con mayor exactitud.

Por otro lado, cuando los TIM son capaces de enviar directamente un JSON sin pasar por el NCAP o con poca injerencia de éste, no es necesario utilizar esta estrategia del punto fijo y la descripción podría ser más textual y legible. No obstante, con la estrategia de enviar más de un campo Ted-id, realmente evitaría las restricciones anteriores, pues se podrían enviar tantos campos Ted-id como sean necesarios o nuevos campos, por ejemplo, semánticos, incluso nuevos protocolos propietarios. El uso de TED virtuales permite usar sensores y actuadores no inteligentes, que son la mayoría del mercado.

4.5. Flujo de datos de sensores y actuadores en la arquitectura propuesta

La figura 4.10 representa el flujo de datos de operación en el sistema desde el punto de vista de un sensor (en la figura 4.10, en la parte superior) y un actuador ((en la figura 4.10, en la parte inferior)). En el caso del sensor, se supone que el sistema lo descubrió previamente, por lo que los metadatos correspondientes se crearon en la base de datos. La figura 4.10 muestra el uso de dos transductores específicos: un sensor analógico y un actuador digital. El sensor analógico se representa con el icono de un termómetro y su flujo de datos describe la publicación del valor de temperatura en tiempo real en el broker MQTT. Dichos valores son consumidos por el mecanismo plug-and-play para actualizar la base de datos de la nube. En la Figura 4.10, un cliente remoto puede leer la temperatura obtenida (23°C) a través de un dashboard.

En cuanto al flujo de datos del actuador, comienza en el dashboard, donde hay un botón para alternar el estado del actuador. Cuando el usuario presiona un botón del dashboard de instrumentos, el sistema publica en el broker MQTT el comando "Turn On". Tal comando es consumido por el motor PnP VTEDS para actualizar la base de datos. El mensaje MQTT también lo consume el NCAP, que lo transfiere al TIM que controla el actuador. Eventualmente, el TIM procesa el mensaje NCAP y activa la salida correspondiente, activando así el actuador deseado.

4.5.1. Flujo de datos de calibración

La tecnología actual de fabricación de sensores proporciona sensores precisos y de alta resolución, pero su precisión y repetibilidad (es decir, idealmente, un sensor siempre debe producir la misma salida para la misma entrada) se ven afectados por ruido, histéresis, linealidad o velocidad de cambio. Por esta razón, en la telemetría es muy importante calibrar los sensores para mejorar la precisión de los datos.

La calibración consiste en un conjunto de operaciones que permite establecer, bajo ciertas condiciones, la relación entre la salida de un sensor y los valores reales detectados

por un dispositivo de medición preciso. Tal relación generalmente se modela como una función, que puede incluir, por ejemplo, factores lineales, exponenciales o logarítmicos. Por lo tanto, la forma principal de establecer esta relación consiste en utilizar tablas de corrección o ecuaciones de calibración (por ejemplo, líneas rectas o curvas). Por ejemplo, en el caso de las líneas rectas, la calibración se puede realizar con la corrección de un solo punto (generalmente compensando el origen) o con dos puntos de calibración, lo que cambia la pendiente de la línea. En el caso de sensores que no siguen una curva de respuesta lineal, el ajuste a menudo se lleva a cabo a través de curvas polinomiales. Se pueden encontrar más detalles y una buena introducción al proceso de calibración en [252].

Como una demostración de las capacidades del sistema propuesto, su primera versión es capaz de hacer uso de ecuaciones de corrección lineal. Sin embargo, sería fácil agregar cualquier otra ecuación o técnica dependiendo de los sensores usados.

La figura 4.11 muestra el diagrama de flujo de datos del procedimiento de calibración. El flujo (secuencia de flujo de datos de calibración ajustada por pasos) comienza cuando un usuario cambia los parámetros de calibración del sensor de temperatura del ejemplo anterior. Dichos parámetros se modifican en los metadatos del sensor en la base de datos y se publican en el tópico MQTT de calibración mediante el motor Pnp VTEDS. El flujo finaliza en el sensor inteligente que está escuchando en el mismo tópico MQTT, para auto-calibrarse. En la próxima lectura del sensor, el valor de la temperatura se ajustará mediante los parámetros de calibración (secuencia de flujo de datos calibrados por lectura de pasos). Por lo tanto, la calibración del sensor se puede realizar de forma remota y automática, evitando la necesidad de llevar a cabo calibraciones in situ por parte de los técnicos y proporcionando flexibilidad y dinamismo al sistema IoT que implementa la arquitectura propuesta.

4.6. Implementación de la arquitectura

La arquitectura y metodología plug and play mostrada anteriormente debe ser implementada sobre tres elementos básicos, TIM, NCAP y Cloud. En esta sección se indicará los elementos esenciales de implementación, de tal manera que puedan ser reproducibles en sus propios proyectos, utilizando dispositivos de hardware y herramientas de software comerciales como propietarias.

4.6.1. Implementación en TIMs

Existen actualmente una gran cantidad de dispositivos comerciales, con diferentes recursos, interfaces de comunicación, CPU, RTOS que pueden actuar como TIM y sobre

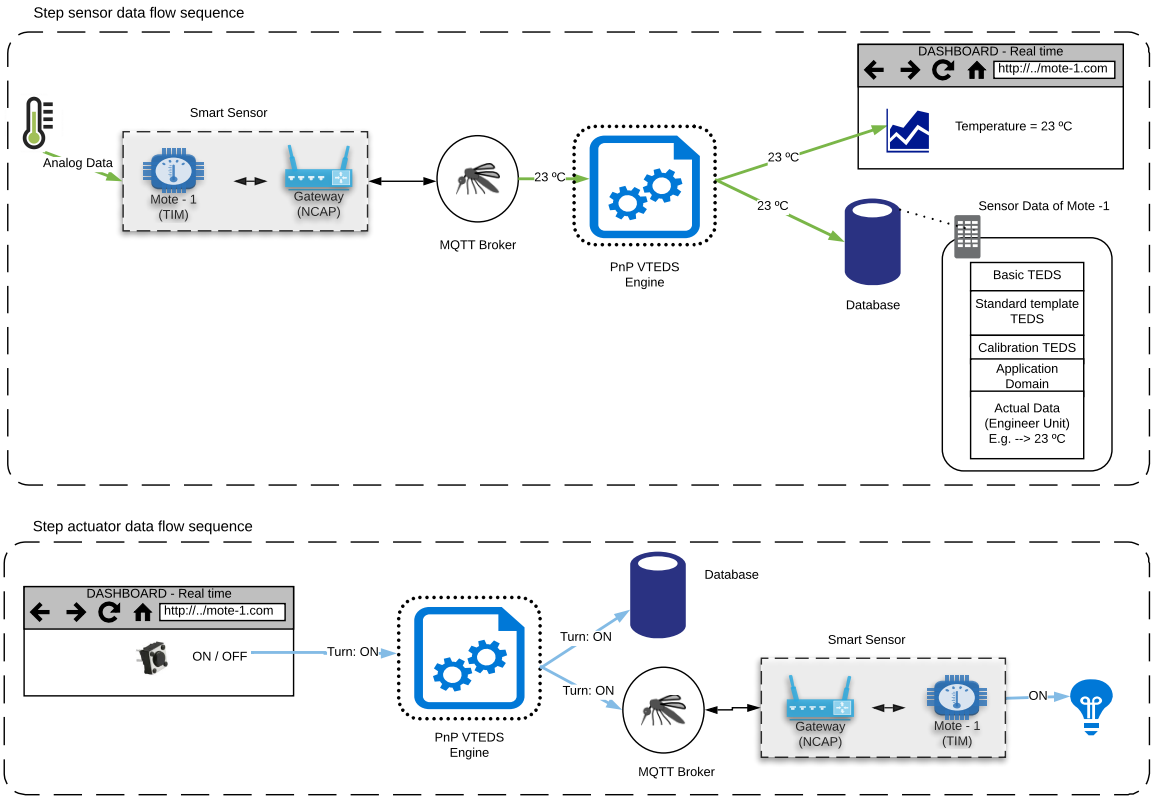


Figura 4.10: Flujo de datos entre sensores/actuadores y las aplicaciones IoT.

los cuales el lector podrá implementar el firmware según los diagramas de flujo mostrados en la figura 4.12. Solo por mencionar algunos de estos dispositivos, se encuentran varias versiones de Arduino con Ethernet shields, varios dispositivos con Bluetooth Low Energy como RedbearLab y nRF52 de Nordic Semiconductor.

El programa principal del TIM, como se aprecia en la figura 4.12 (a), debe incluir funciones que permitan primeramente descubrir el hardware instalado. Se recomienda el uso intuitivo y sencillo de etiquetas, pero cualquier otra técnica es válida también. Esto permitirá al TIM auto configurarse y estar listo para gestionar los sensores y actuadores presentes en sus entradas y salidas digitales y analógicas. Además, el firmware debe incluir funciones que permitan crear los meta-descriptores en función de la potencialidad del hardware usado. En caso de dispositivos restringidos en memoria y CPU se recomienda utilizar la metodología propuesta para representar el TED-id a través de un número real representado en punto fijo 8.8. Luego, el TIM debe enviar un frame de configuración a las capas superiores del sistema, dígase NCAP y Cloud para lograr auto-registrarse. Una vez conseguido esto, los sensores del TIM podrán ser monitoreados remotamente desde la web, y capaz de recibir comandos remotos para sus actuadores.

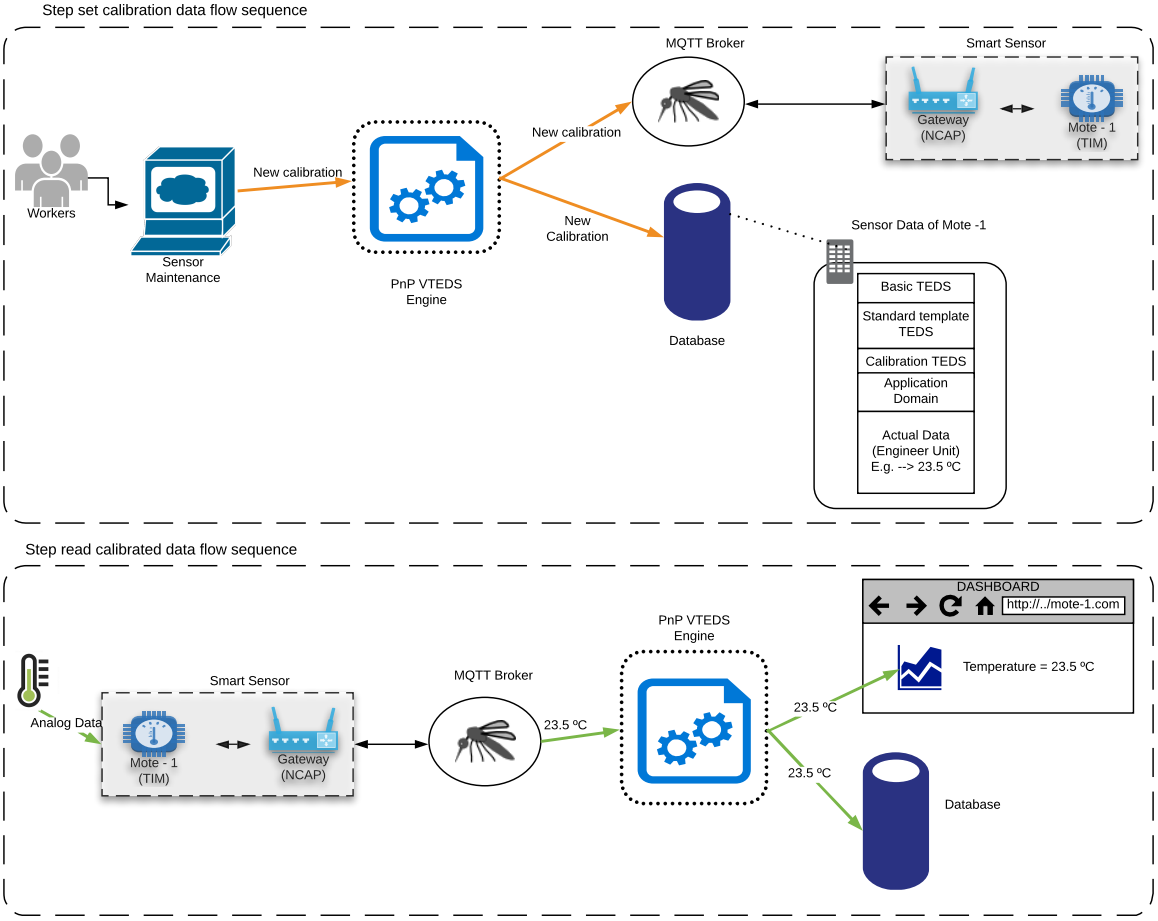


Figura 4.11: Flujo de datos para nuevas calibraciones de sensores.

Se recomienda gestionar a los sensores y actuadores a través de subrutinas de interrupción para una mayor eficiencia en el desempeño de los TIM. La figura 4.12 (b) muestra las ISR que deben ser implementadas para una gestión en tiempo real. Otros ejemplos de implementación de nuestra metodología en un TIM basado en Eddystone beacon con altas restricciones, incluso en la cantidad de bytes disponibles a transmitir, pueden ser encontrados en [124].

4.6.2. Implementación en NCAP

El otro elemento importante del mecanismo plug and play lo constituye el NCAP, el cual y como hemos planteado a lo largo de este capítulo no solo actúa como el gateway de la WSN sino que realiza funciones de FOG computing, agregando valor al sistema IoT. En dependencia de los TIM usados, medios de comunicación, potencia de procesamiento, memoria se debe seleccionar el hardware para el NCAP. Aquí también pueden ser usados tanto versiones comerciales como propietarias. Algunas de los NCAP

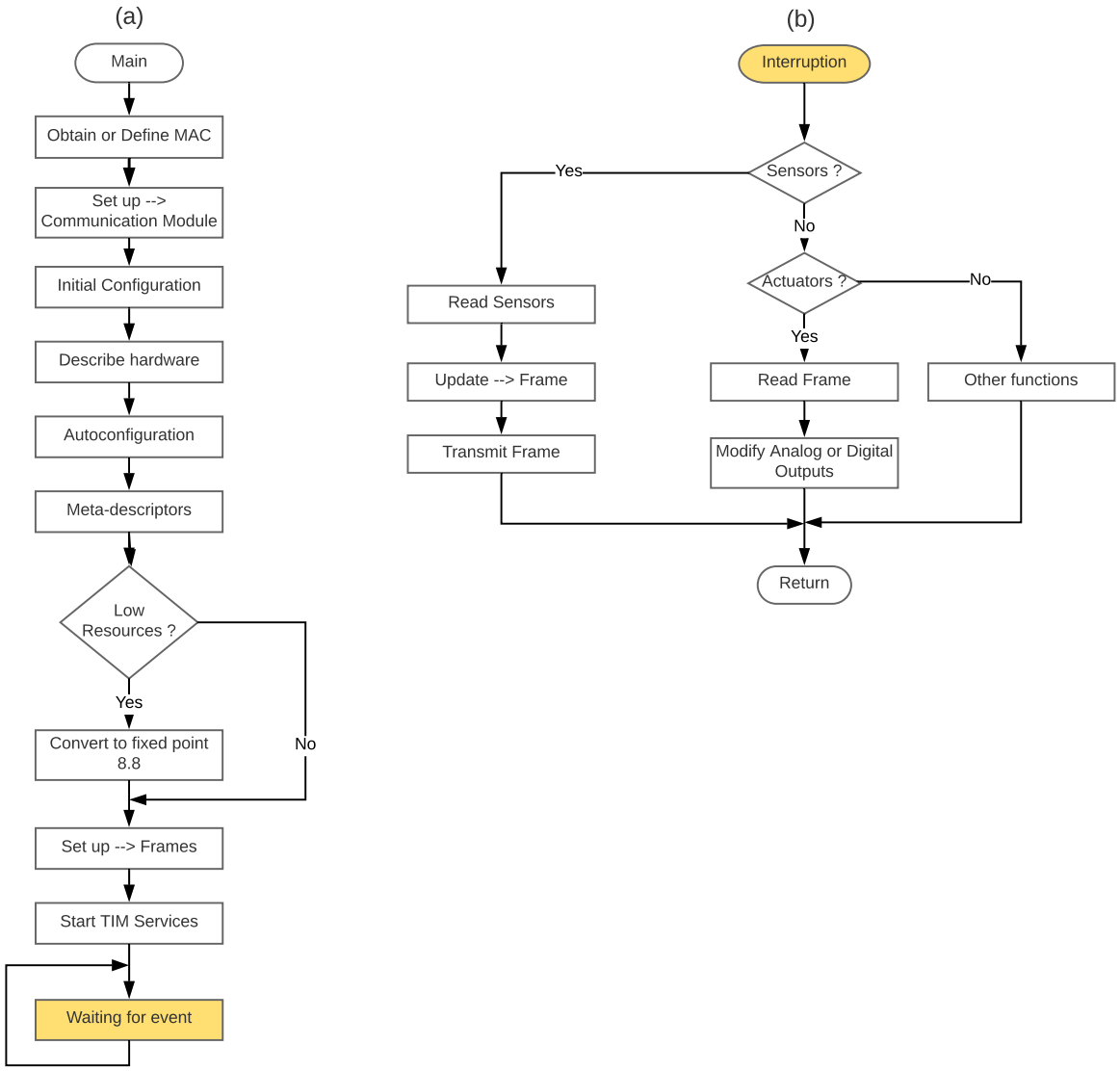


Figura 4.12: Diagramas de flujo para un TIM. (a) Main y (b) Interrupt Service Routine (ISR).

comerciales que hemos implementado están las diferentes versiones de Raspberry Pi, Beaglebone, Orange Pi y smartphones con Android OS.

El grado de complejidad de implementación de un NCAP puede verse incrementado exponencialmente en función del hardware seleccionado, dominio de aplicación IoT, visualización local, base de datos, Fog computing, medios de comunicación entre otras. No obstante, la figura 4.13 podría ser un excelente punto de partida para el desarrollo y programación del NCAP. En este caso, se muestra una máquina de estados resumida de un NCAP, implementado sobre un smartphone con una aplicación Android para gestionar TIM basados en beacons and characteristics Bluetooth Low Energy. Con algunas funciones añadidas de FOG computing para visualizar en tiempo real valores

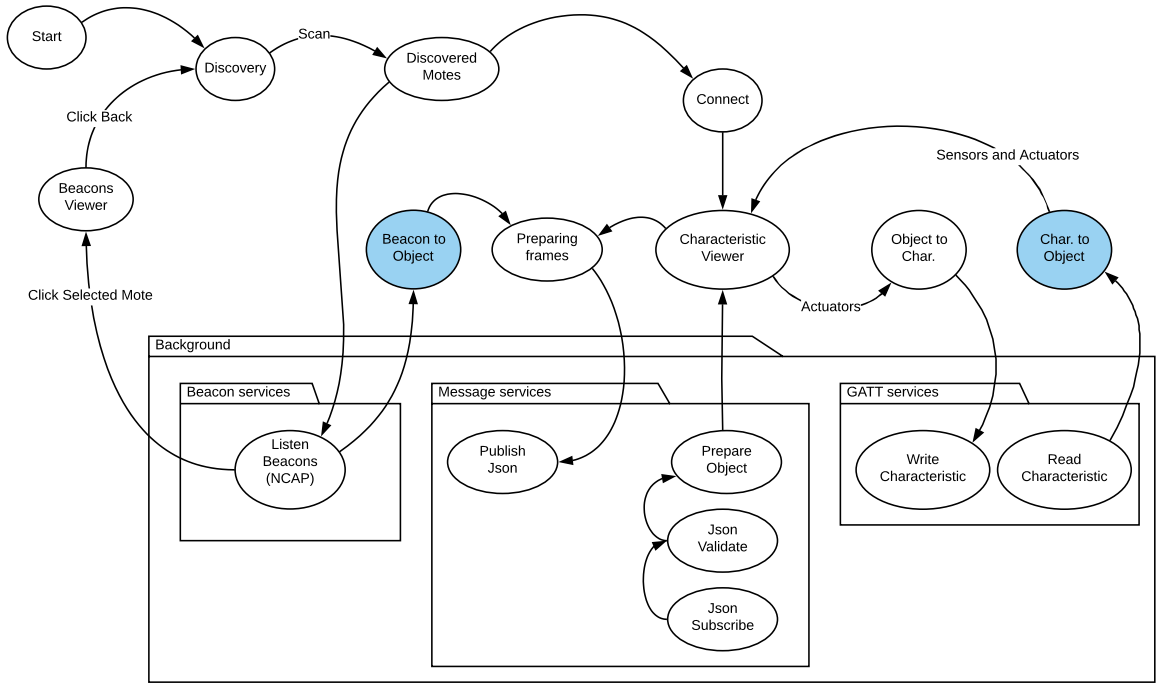


Figura 4.13: Máquina de estados para implementar un NCAP.

de sensores y enviar comandos localmente a los actuadores del TIM seleccionado. Es importante destacar que en los estados resaltados de la figura 4.13 (Beacon to Object and Characteristics to Object) se deben incluir funciones para la gestión de los metadatos del TIM, por ejemplo, los TED-id del TIM descubierto y seleccionado. Un ejemplo de implementación de este NCAP puede ser encontrado en [253].

4.6.3. Implementación de la cloud IotM@ch

El tercer elemento importante para la implementación de nuestra arquitectura plug and play basada en VTEDS constituye precisamente la cloud, que es donde van a residir y virtualizar los TEDS, que ampliamente hemos detallado en secciones anteriores. La cloud al igual que los otros dos elementos expuestos, pueden ser comerciales o propietarios. Entre las comerciales más conocidas tenemos a Amazon AWS [81], Microsoft Azure [254] y Google Cloud platform [83]. En nuestro caso decidimos utilizar la cloud desarrollada por el grupo de investigación AutoMathTIC de la Universidad Técnica de Machala (Ecuador) en conjunto con la Universidade da Coruña (España), llamada IotM@ch cloud [92] diseñada a partir con las herramientas open source propuestas en [195].

La plataforma proporciona funciones y funcionalidades similares a otras plataformas comerciales de IoT, por lo que se seleccionó principalmente debido a dos factores: disponibilidad y costo. Por un lado, la plataforma IotM@ch era completamente accesible,

con control total sobre el hardware y el software de la infraestructura de la nube. También es posible administrar el ancho de banda del servidor local, lo que permite aumentar el rendimiento y reducir la latencia de la aplicación para los clientes remotos de Internet. Por otro lado, la Universidad Técnica de Machala ya había instalado la infraestructura del servidor, con la posibilidad de utilizar los recursos de la universidad sin incurrir en costos adicionales de hardware y sin necesidad de contratar y entrenar técnicos dedicados a administrar servidores, control de acceso, aire acondicionado, baterías o subsistemas de respaldo de energía. Además, tenga en cuenta que, debido a la cantidad de servicios y aplicaciones desplegados por el sistema propuesto, que requieren el uso de 5 a 8 máquinas virtuales, el costo mensual de otras plataformas IoT como Google IoT generalmente oscila entre $ 2,000 y $ 3,000, lo que es prohibitivo para muchas universidades en países en desarrollo. Por ejemplo, se obtuvo un costo mensual estimado de $ 2,372.35 a través de Google Cloud Platform Pricing Calculator [255] con los siguientes parámetros: Motor informático: 1x Análisis de datos (n1-estándar-4); Funcionamiento, sistema / software: gratuito (Debian, CentOS, CoreOS, Ubuntu u otro SO de proveedor de usuario), 1x Big data (24 horas por día), 1x APP Server (n1-estándar-2, SSD space 2x375 GB) , 1x Monitoreo en tiempo real (n1-standard-2), almacenamiento en la nube (4,096 GB), Cloud SQL de segunda generación (una instancia, 500 GB de almacenamiento, 500 GB para copias de seguridad) y IoT Core (datos intercambiado: 102,400 MB).

La figura 4.14 muestra los aspectos más relevantes de dicha cloud, resaltando la presencia de la aplicación web de Monitoreo y control residente en la máquina virtual 3. Esta aplicación permitirá la gestión tanto de los nodos sensores descubiertos dinámicamente como el servidor de los virtual TEDS. Además, dispone de funciones para generar reportes a partir de los datos históricos como la visualización en tiempo real de sensores y actuadores a través de dashboards generados dinámicamente con la información actual de los nodos sensores que se auto registraron en el sistema IoT.

En las próximas secciones se mostrarán las principales interfaces de la aplicación mencionada, explicando la funcionalidad de las mismas.

4.6.3.1. Gestión de TEDS

La figura 4.15 muestra una visión panorámica de la aplicación web multi-idioma desarrollada. En la columna izquierda de la página podemos ver el listado de funciones implementadas para la gestión de los parámetros necesarios de la arquitectura plug and play diseñada. Más adelante serán mostradas estas funciones. En la parte derecha de la imagen se pueden ver un conjunto de metadatos de sensores y actuadores. Desde ésta propia interfaz el usuario puede editarlos o eliminarlos.

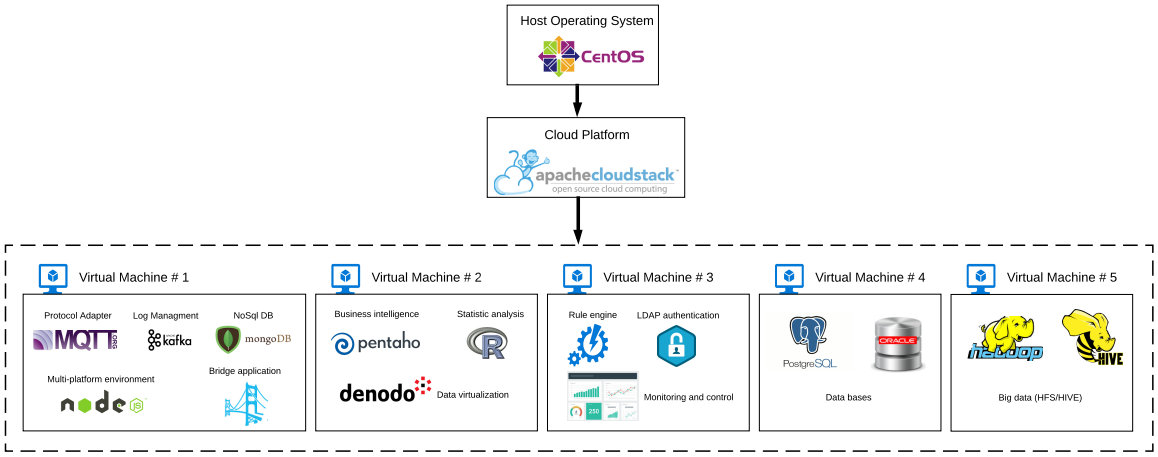


Figura 4.14: Estrucutra resumida de la Cloud IotM@ch.

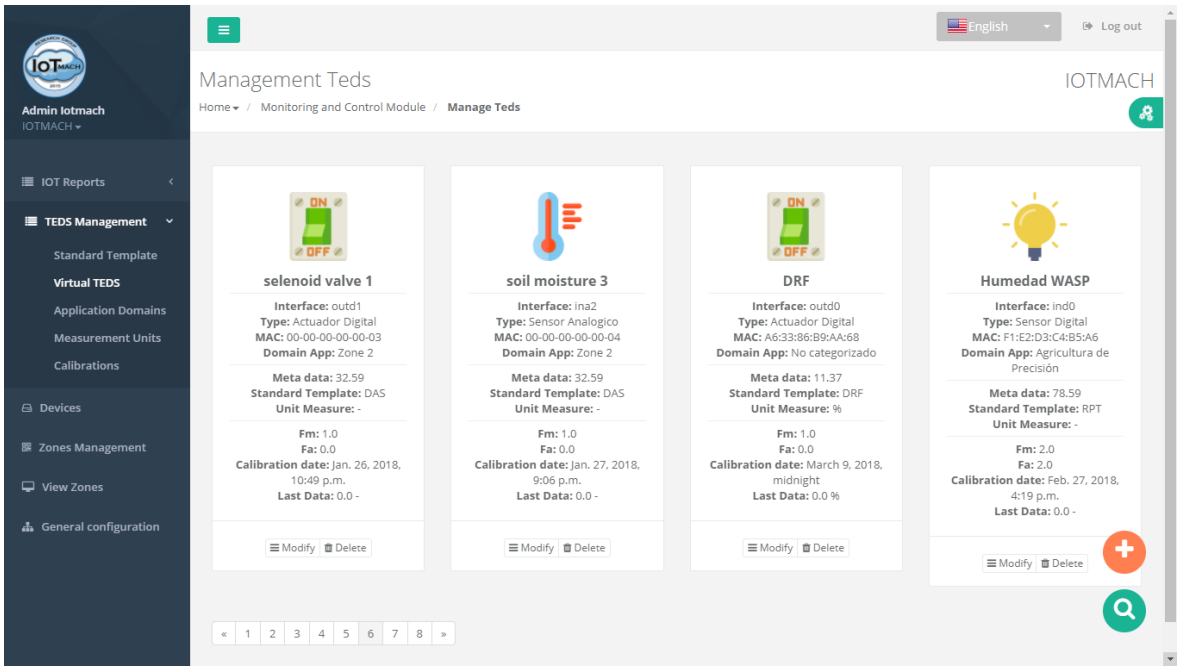


Figura 4.15: Vista principal para gestionar los TEDS.

Standard template

La figura 4.16 muestra el interfaz necesario para gestionar el standard template mencionado en el estado del arte de este capítulo. Desde aquí, se pueden añadir también nuevos templates en función de la necesidad del usuario.

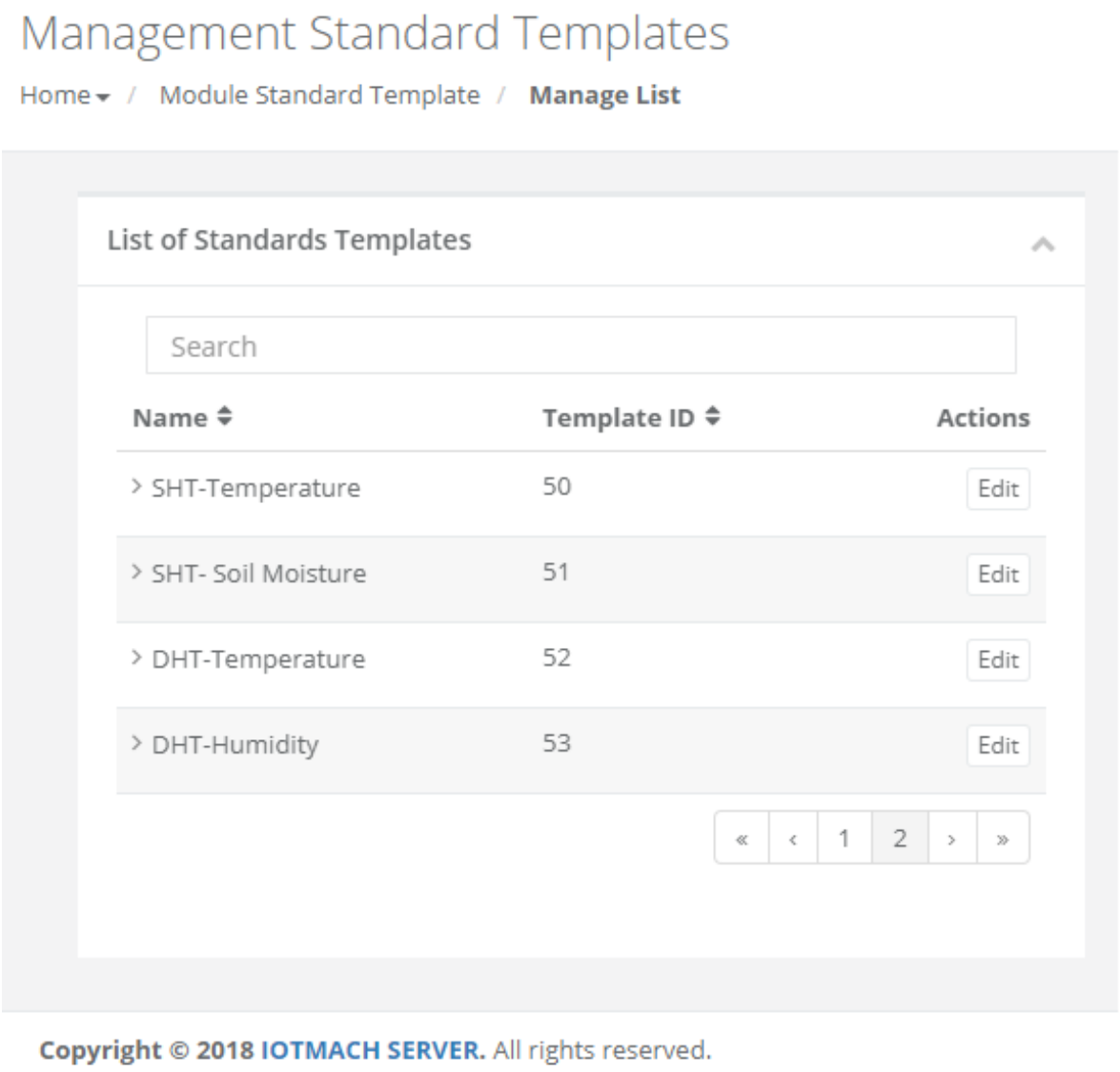


Figura 4.16: Standard templates.

Virtual TEDS

En el extremo izquierdo de la figura 4.17 muestra de forma ampliada el contenido de uno de los virtual TEDS listado en la figura 4.15, con todos los descriptores enviados por el TIM, es decir la información mostrada no es estática, incluso puede ser editada manualmente como se observa en el diálogo presente en el lado derecho de la imagen.

Es interesante destacar que en este diálogo existe la posibilidad de crear manualmente un TED-id. Observe que el campo metadata se genera automáticamente a partir de los elementos seleccionados en los campos standard template y unidad de medida.

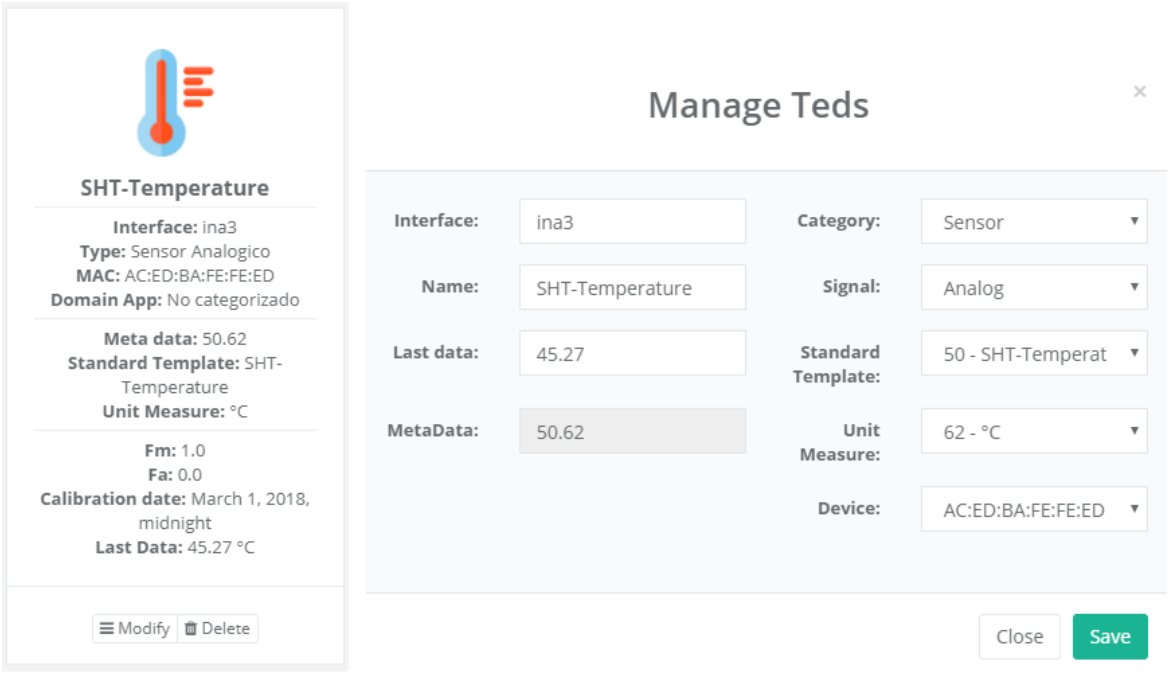


Figura 4.17: TEDS virtuales.

Dominios de aplicación

La figura 4.18 muestra el interfaz disponible para crear manualmente los diferentes dominios de aplicación. En esta misma interfaz se listan los dominios de aplicación descubiertos a través de las tramas de configuración de los TIM en el proceso de auto registro en la cloud.

Unidades de medida

La figura 4.19, muestra el interfaz para gestionar todos los tipos de unidades de medida que fuesen necesarios, en función de los sensores y actuadores disponibles en el mercado o propietarios.

Calibraciones

Las calibraciones de los sensores y actuadores en nuestra arquitectura ocurren de forma dinámica como ya fue explicado en secciones anteriores. La figura 4.20, muestra el interfaz a través del cual un técnico de mantenimiento puede añadir nuevos parámetros de configuración de unos de los sensores listados (descubiertos). Una vez que se presiona el botón de calibrar, el sistema envía un mensaje MQTT con la información necesaria, consumida luego por el NCAP correspondiente. También de está propia interfaz

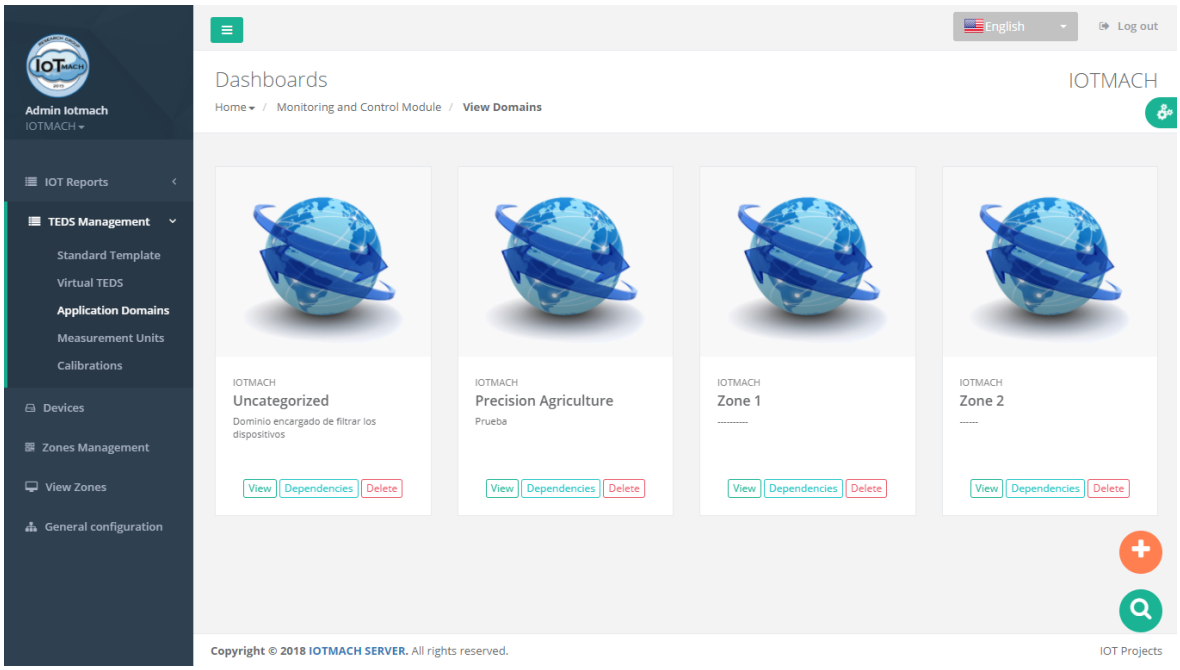


Figura 4.18: Dominios de aplicación.

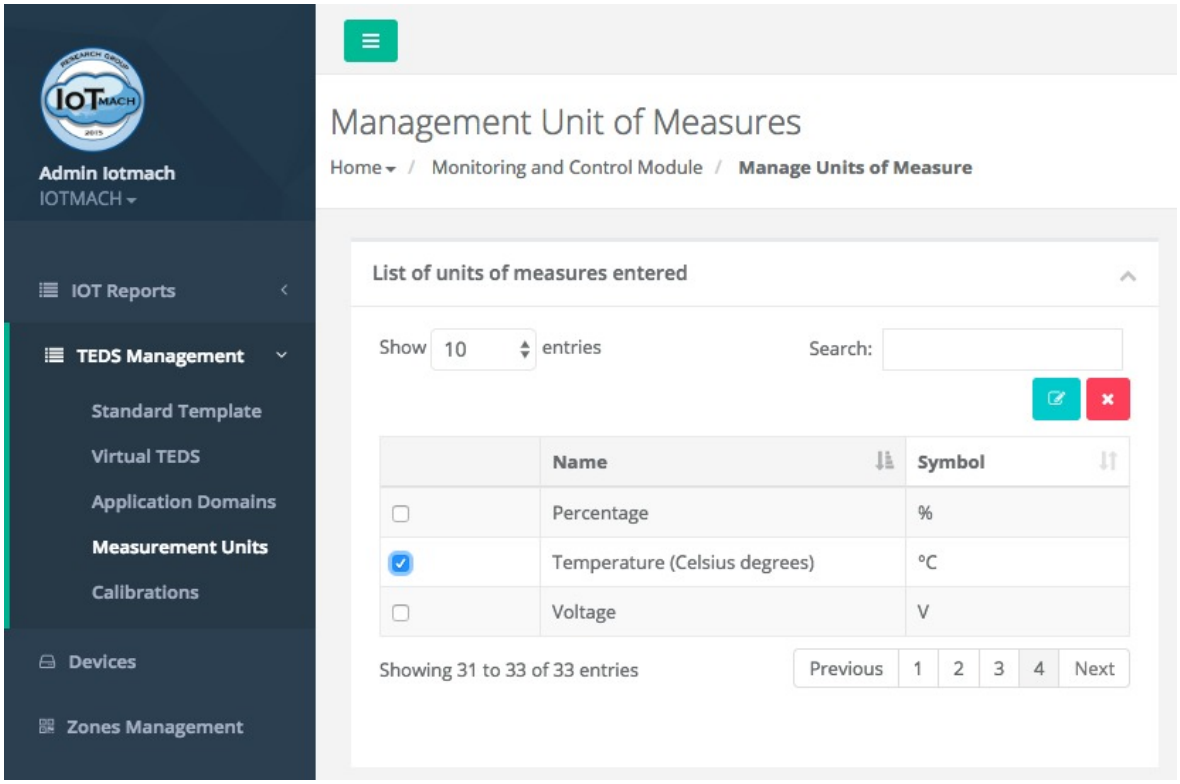


Figura 4.19: Unidades de ingeniería.

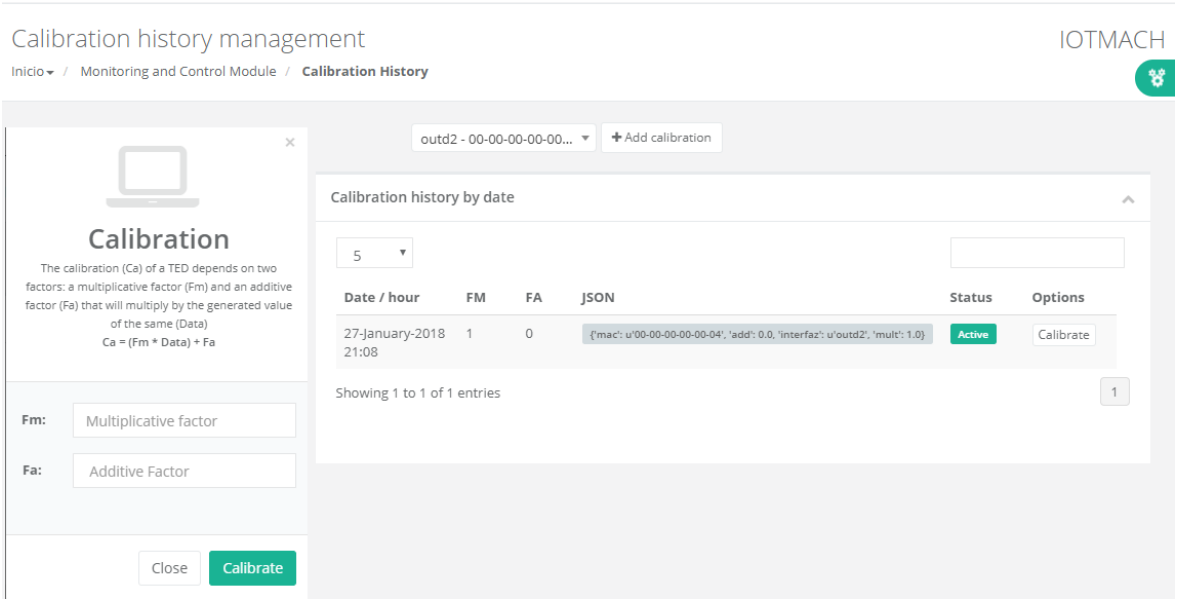


Figura 4.20: Interfaz gráfica para gestionar las calibraciones.

se pueden acceder a calibraciones históricas y tomar dichos parámetros y enviarlos nuevamente al sensor requerido.

4.6.3.2. Gestión de Dispositivos

Todos los nodos sensores descubiertos por la cloud, ya que realizaron el proceso de auto registro correctamente, son listados a través del interfaz de la figura 4.21. Una vez que ya existen en la cloud, pueden ser editados manualmente. Por ejemplo, para añadir una imagen distintiva para una mejor representación visual.

4.6.3.3. Gestión de Dashboards

Como ya hemos comentado anteriormente, la arquitectura plug and play implementada según la metodología diseñada, permite generar dashboard dinámicamente, es decir el contenido a mostrar de un nodo sensor, será aquel que el TIM informó a través de las tramas de configuración en el proceso de auto registro. No obstante, la aplicación web permite personalizar la información que se desea visualizar en un dashboard. Esto permite mostrar de forma conjunta la información disponible de diferentes sensores, incluso de diferentes TIM.

Para conseguir esto, se dispone de la interfaz mostrada en la figura 4.22. Desde la cual se pueden gestionar diferentes zonas de visualización pudiendo escoger los sensores y actuadores listados en el interfaz de Device Management, mostrado anteriormente.

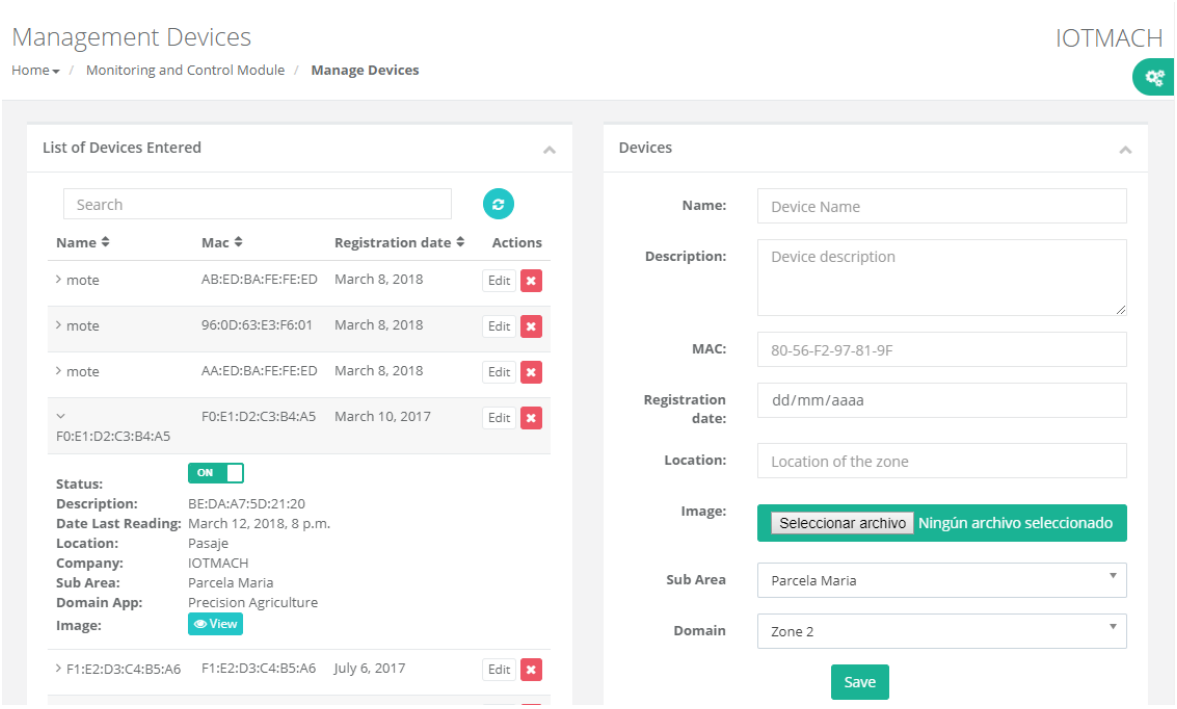


Figura 4.21: Dispositivos descubiertos.

Una vez que hemos configurado dichas zonas, éstas serán listadas y mostradas en una nueva interfaz, mostrada en la figura 4.23. Donde el usuario puede seleccionar uno de los dashboard ya configurados. La figura 4.24, muestra un ejemplo de un dashboard generado dinámicamente con información en tiempo real de sensores y actuadores.

4.6.3.4. Visión General

Para una mejor visualización y gestión de todos los elementos descubiertos dinámicamente en la cloud, se dispone de una estructura en forma de árbol desplegable que permite acceder de forma inmediata a la información requerida, como se aprecia en la figura 4.25.

4.6.3.5. Gestión de reportes

Como hemos podido observar en las imágenes anteriores, la aplicación permite la visualización en tiempo real de sensores y estado de los actuadores de todos los TIM descubiertos. Pero, es importante también disponer de una interfaz para generar reportes a partir de la información guardada en la base de datos. La figura 4.26 muestra el interfaz que permite al usuario personalizar la información deseada en el reporte.

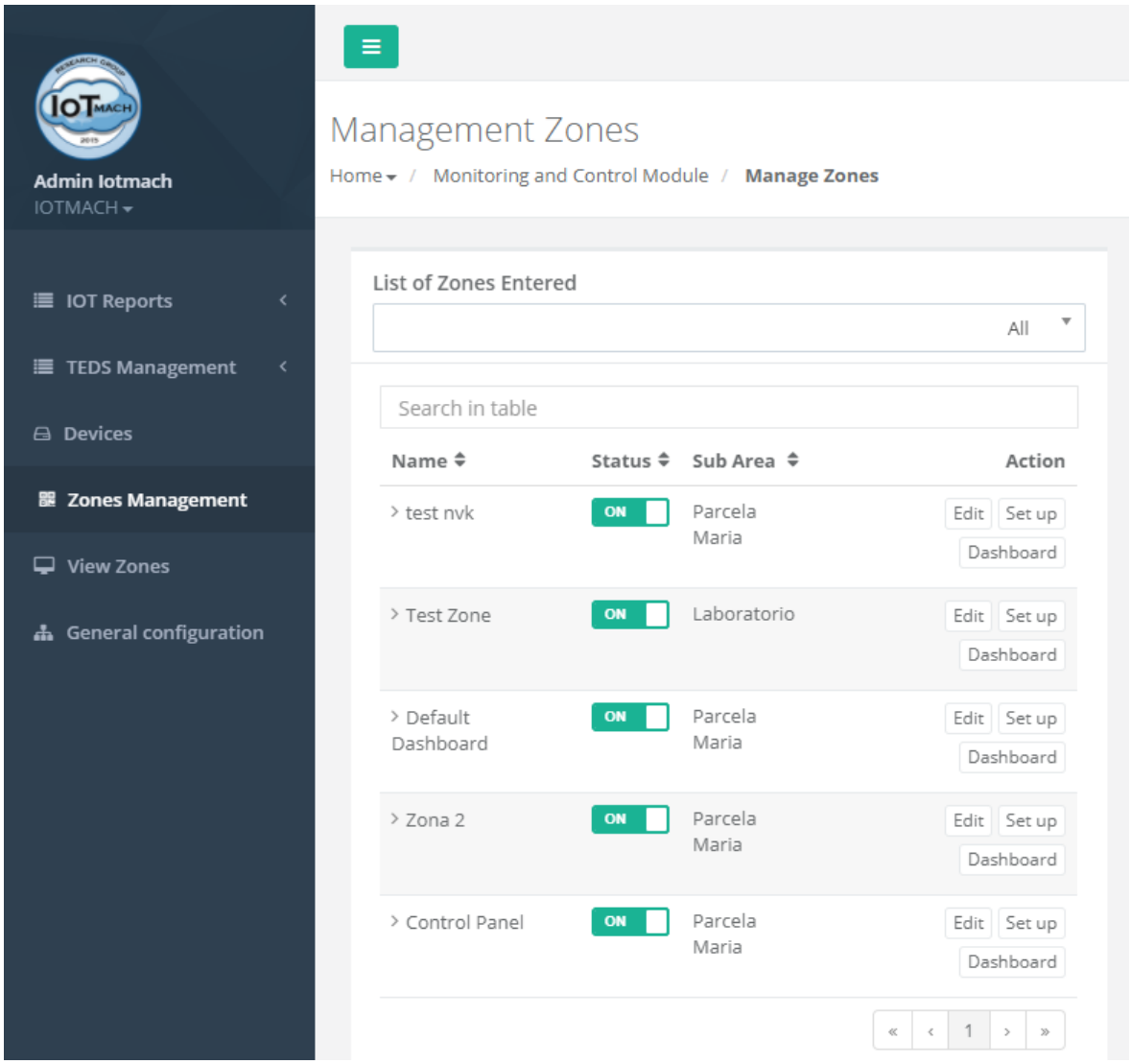


Figura 4.22: Gestión de zonas de visualización.

4.7. Experimentos

En esta sección la arquitectura propuesta fue puesta a prueba en un ambiente IoT global real, en este caso la cloud se encuentra en la Universidad Técnica de Machala, Ecuador y los nodos sensores (TIM and NCAP) se encontraban en la Universidad da Coruña, España, tal y como se puede apreciar en la figura 4.27.

Estos experimentos permitirán poner a prueba al sistema diseñado, basado en protocolos ligeros y virtual TEDS. La comparación fue llevada a cabo midiendo los tiempos para auto-configuración (self-configuration), auto-registro (self-register), telemetría (telemetry) y auto-calibración (auto-calibration), que permiten evaluar el impacto real de la latencia en el rendimiento general de la arquitectura plug-and-play con tres

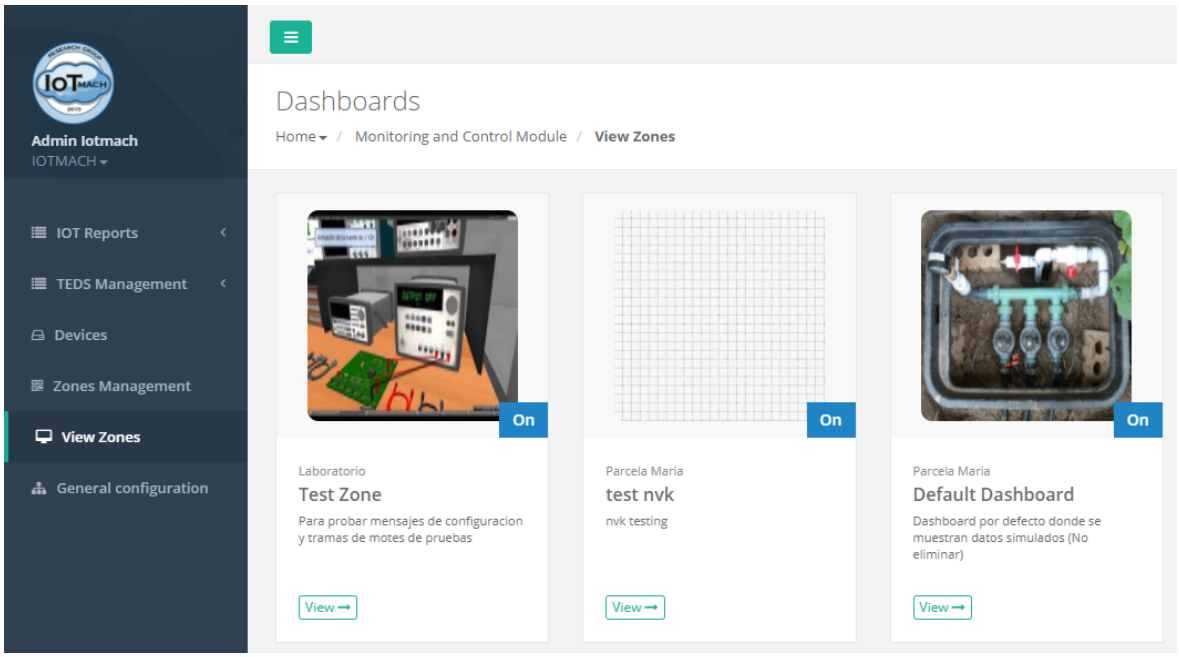


Figura 4.23: Listado de zonas de visualización.

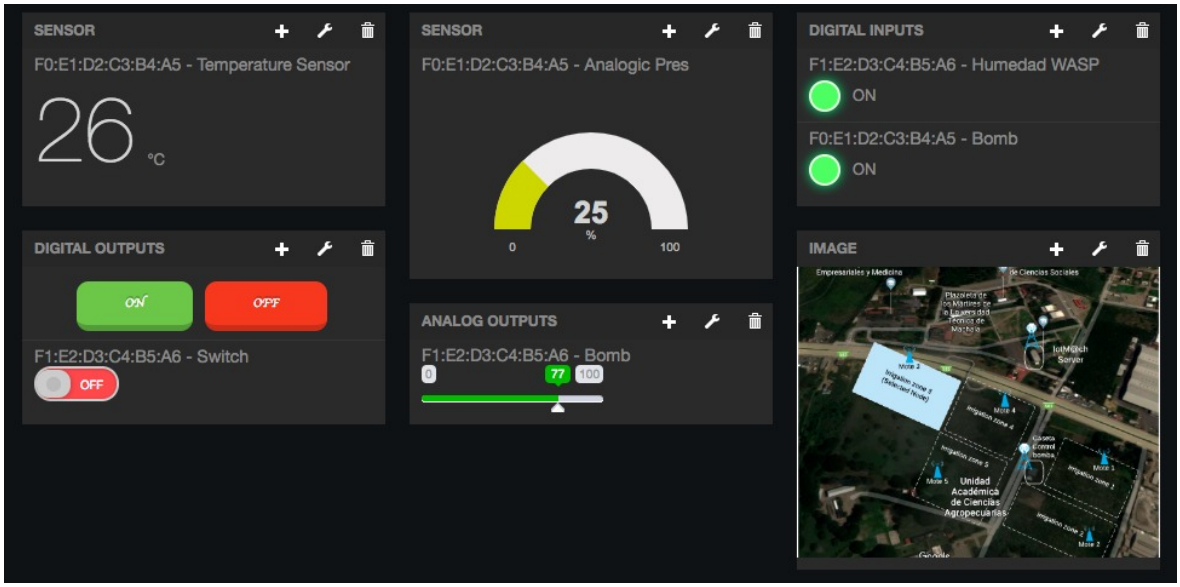


Figura 4.24: Gestión de zonas de visualización.

TIM diferentes: un beacon Eddystone BLE, un beacon basado en BLE GATT y un microcontrolador con una interfaz Ethernet.

En el caso de los beacons BLE, tenga en cuenta que se trata de pequeños dispositivos livianos de bajo consumo y bajo costo que transmiten ciertos paquetes de información periódicamente para indicar su presencia o transmitir ciertos datos. Por ejemplo, se han utilizado tradicionalmente para proporcionar servicios basados en la ubicación. El BLE

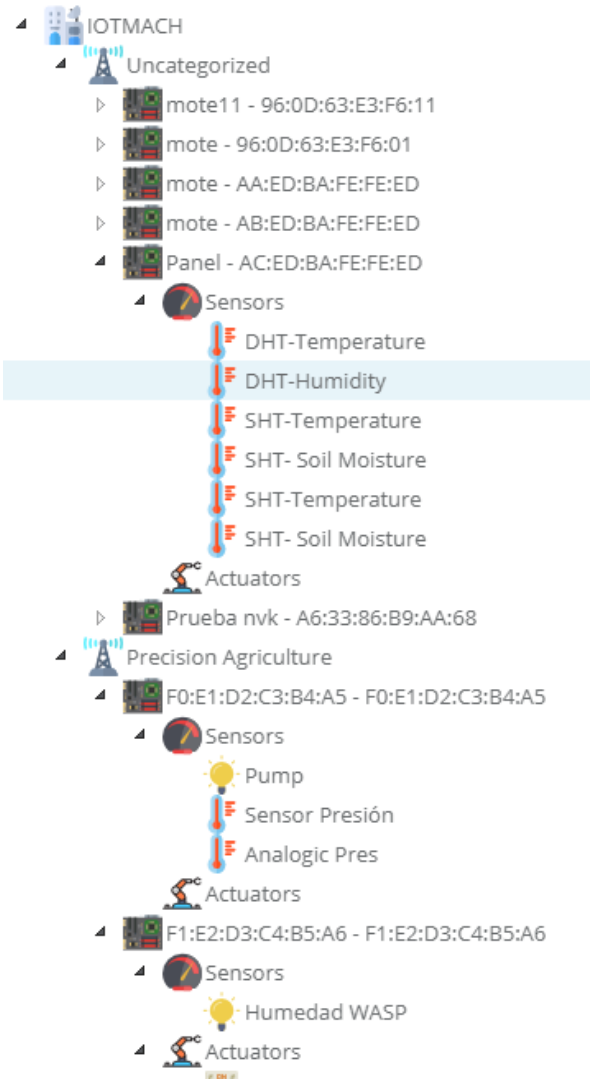


Figura 4.25: Vista general de los nodos sensores en forma de árbol.

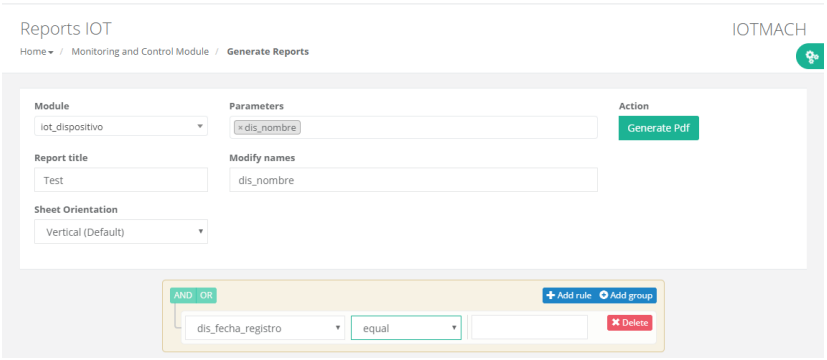


Figura 4.26: Vista general de los nodos sensores en forma de árbol.

Eddystone beacon [123] está basado en Eddystone, el formato BLE beacon de Google. El beacon utilizado en los experimentos se configuró con un intervalo de balizamiento de 250 ms, ya que se determinó en [124] que ofrecía la mejor compensación entre la latencia y el consumo cuando existía la línea de vista (LoS) entre el beacon y un teléfono inteligente, y una distancia de 3 metros entre ambos.

Con respecto al beacon basado en BLE GATT, debe observarse que el beacon actúa como un servidor que proporciona servicios e indica ciertas características (el GATT define la jerarquía y el formato utilizado para representar y alterar los datos BLE). Por lo tanto, para comunicarse con otro dispositivo BLE, se requiere establecer primero una conexión (en contraste, los beacons Eddystone no tienen conexión). Una revisión en profundidad del estándar BLE está fuera del alcance de este documento, pero el lector interesado puede encontrar más detalles sobre BLE y GATT en [172].

Finalmente, con respecto al Ethernet TIM, se debe enfatizar que se seleccionó para determinar el rendimiento de la arquitectura cuando se usa un dispositivo de IoT de restricción de recursos que usa una interfaz cableada. Para las pruebas realizadas, el Ethernet TIM consistió básicamente en un Arduino Mega con una shield Ethernet.

4.7.1. Preparación de los experimentos

La figura 4.28 muestra los elementos utilizados en los experimentos: un Nordic nRF51 development kit (nRF51-DK) que actúa como TIM BLE, tanto para un beacon Eddystone como para el servidor GATT, un teléfono inteligente Android como NCAP y

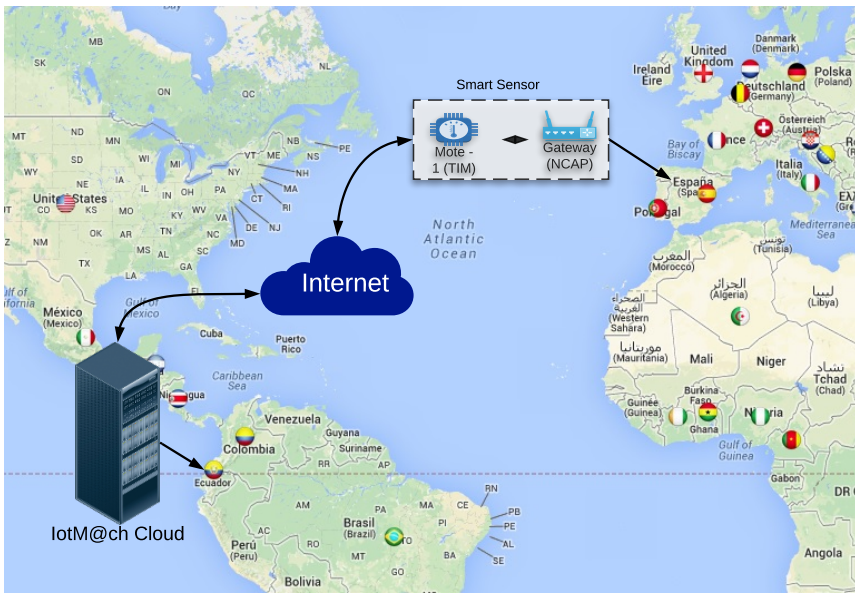


Figura 4.27: Escenario IoT de los experimentos.

una laptop ejecutando los programas Android Studio [256], Serial Port Monitor [257] and Navicat Premium [258]. Las principales especificaciones de todos los elementos se describen en la tabla 4.3.

4.7.2. Medición de latencias

La latencia es la métrica por excelencia para medir el rendimiento en muchas aplicaciones, sobre todo del ámbito del Internet de las cosas. En nuestro caso usaremos esta métrica para conocer la velocidad con la que los nodos sensores son capaces de detectar y habilitar primeramente su propio hardware (Self-configuration). Seguidamente, conocer la velocidad de estos sensores inteligentes para ser detectados (Self-register) tanto a nivel NCAP como la cloud, con mecanismos plug and play. A partir de este momento tanto el sistema IoT como usuarios externos pueden interactuar con los sensores y actuadores descubiertos. Otra latencia medida (Telemetry) nos permite conocer el tiempo que necesita el sistema o un usuario para disponer de la sensor data, desde que el sensor tiene la nueva medición hasta que la data ya en unidades de ingeniería se encuentre

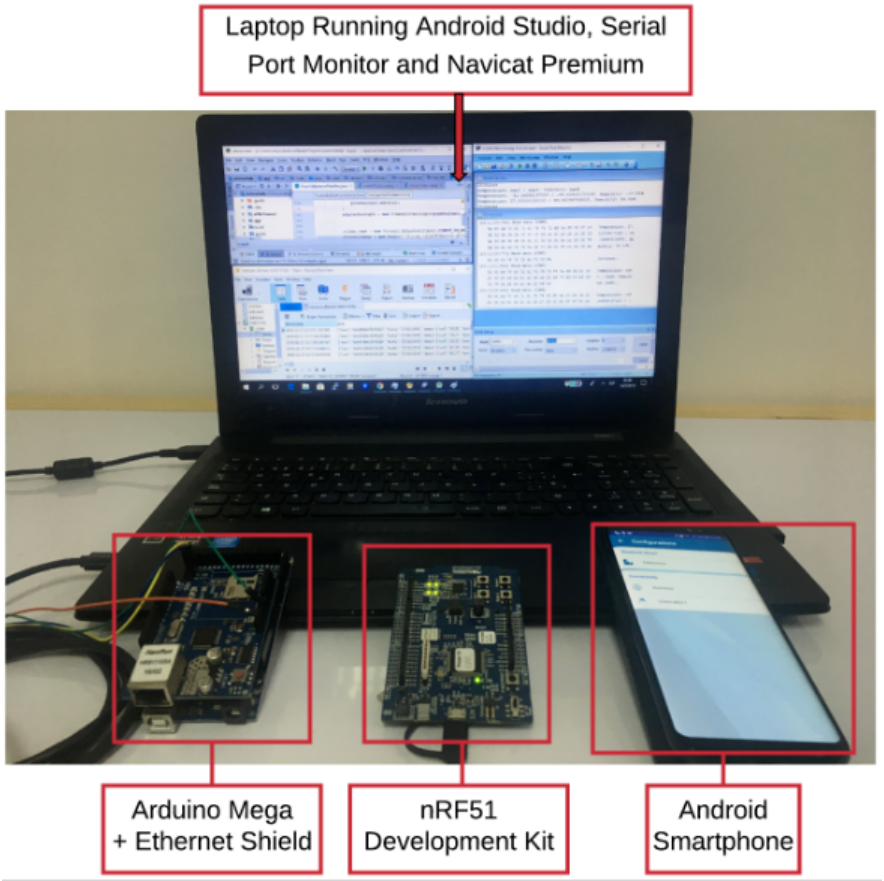


Figura 4.28: Elementos usados en los experimentos.

Tabla 4.3: Especificaciones principales de los elementos usados en los experimentos de VTED.

Elementos usados	Especificaciones principales
Laptop	Lenovo 80E502A5SP G50-80: 15.6", 16 GB RAM, 1 TB HDD Intel® Core™i7-5500U CPU @ 2.40 GHz OS version: Windows 10-Pro x 64 bits
Android Studio 3.0.1	AI-171.4443003, built on November 9, 2017 JRE: 1.8.0.152-release-915 amd64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Nordic nRF51 development kit (nRF51-DK)	SoC: nRF51822, 2.4 GHz multi-protocol device, 32-bit ARM® Cortex™M0 CPU with 256 kB/128 kB flash + 32 kB/16 kB RAM
Smartphone	Samsung Galaxy S-8, 4 GB RAM, 64 GB (UFS 2.1) ROM Model: SM-G950F Android version: 7.0 (Nougat) Processor: Exynos 8895, 2.3 GHz Quad + 1.7 GHz Quad, 8 Cores (Octa-Core)
Arduino Mega	SoC:ATmega2560, 8 bits, 16Mhz Digital I/O Pins: 54, Analog Input Pins: 6 256 KB flash, 8 KB SRAM and 4 KB EEPROM
Ethernet Shield	Ethernet Controller: W5500 with internal 32K buffer, 10/100Mb connection with Arduino on SPI port. Operating voltage 5V (supplied from the Arduino Board)
Serial Port Monitor	Version 6.0, Build 6.0.235 Eltima Software Analyze serial port activity and Monitor several ports within one session.
Navicat Premium	Version 11.0.8, Seamless Data Migration, Diversified Manipulation Tool Easy SQL Editing, Intelligent Database Designer, Advanced Secure Connection Connect to MySQL, MariaDB, SQL Server, Oracle, PostgreSQL, and SQLite

disponible en el NCAP y la cloud. La última métrica de latencia conseguida es la de Auto-calibration, medida desde que un usuario ingresa nuevos parámetros de ajuste hasta que la nueva medición del sensor llegue ya calibrada.

4.7.2.1. Latencia de auto-configuración (Self-configuration)

Para obtener la latencia de self-configuration, los firmwares de los TIM usados en los experimentos fueron modificados para poder calcular internamente dicha latencia. Esto fue necesario ya que estos tiempos son muy pequeños y cuando se utiliza un software para espiar (sniffer) el puerto USB para obtener el timestamp de las etiquetas usadas para los experimentos, el buffer del puerto serial retenía dichas etiquetas y cuando se mostraban en el sniffer las marcas de inicio y fin tenían el mismo timestamp. La figura 4.29 muestra el procedimiento empleado, basado en el diagrama de flujo ya presentado en la figura 4.12. La estrategia consiste en capturar el ticker del RTOS utilizado al inicio y al final de la ejecución de las funciones requeridas por el TIM para descubrir y habilitar el hardware presente con el reinicio del CPU. La diferencia entre ellas sería la latencia deseada y puede ser enviada a través de su puerto serial y capturada por cualquier sniffer USB, como pueden ser Serial Port Monitor o el Tera term [259] que fue el utilizado para estas pruebas como se puede apreciar en la figura 4.29. Donde la función *us_ticker_read()* presente en los cuadros resaltados del diagrama de flujo,

corresponde a la función usada en el IDE online m-bed para leer el contador de ticker del CPU empleado. En nuestro caso el firmware de los TIM BLE (beacon y GATT) fueron desarrollados con m-bed. El ejemplo de dicha figura muestra una latencia de self-configuration de 133.301 ms.

Una vez re-programados los TIMs, el experimento consiste en apagar y encender varias veces el TIM y obtener del programa Tera term el valor de la latencia promedio. En la tabla 4.4 se muestran los resultados del promedio de 50 ensayos realizados para los tres TIM envueltos en la investigación, un Eddystone beacon, un GATT server y Arduino con Ethernet shield. Es importante indicar esta latencia aparece solo al inicio del firmware del TIM con cada reinicio del CPU, por tanto, para TIM estáticos o sin caídas de alimentación está latencia no afecta el rendimiento global del sistema u otras latencias. No obstante, y como se verá más adelante influye considerablemente en las

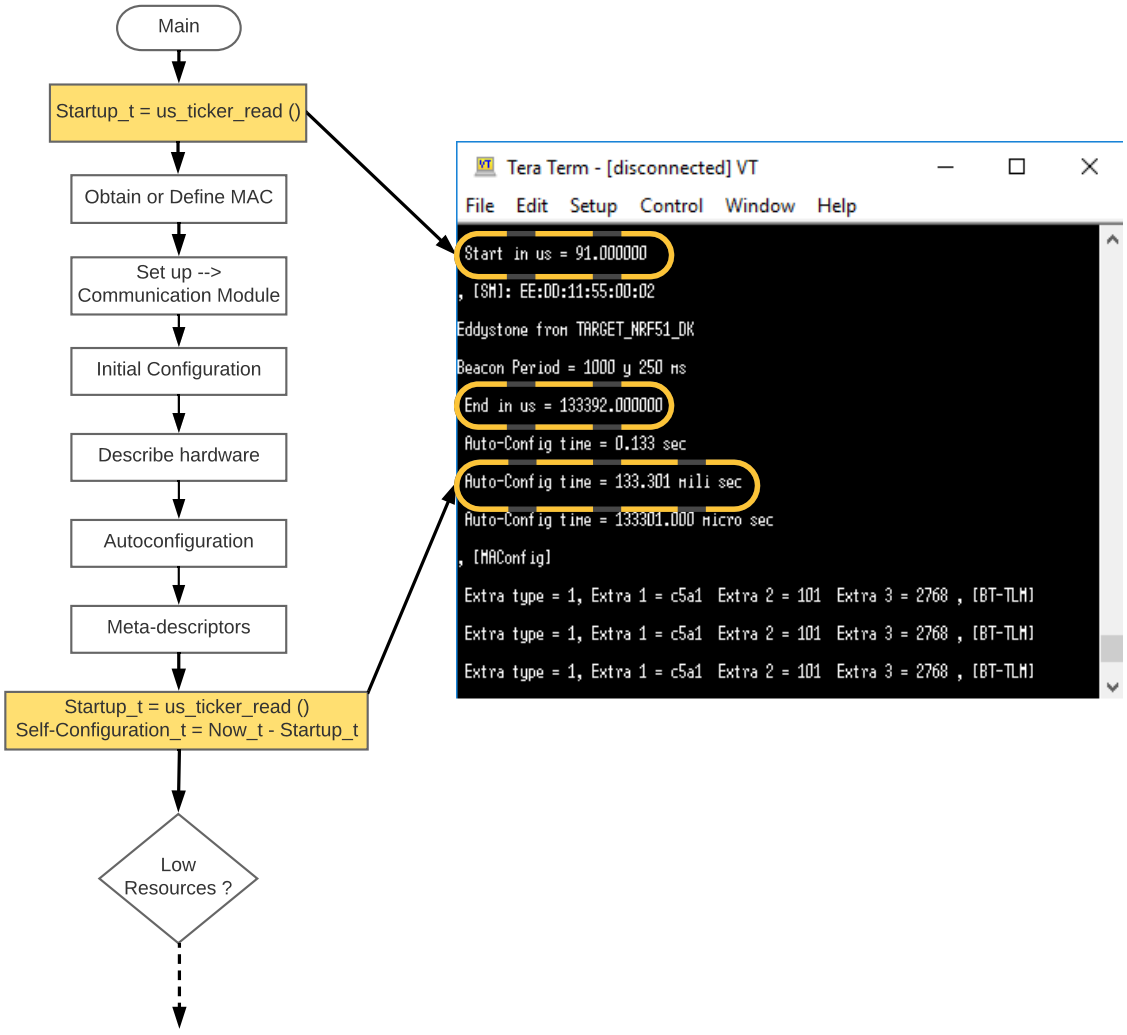


Figura 4.29: Metodología para determinar la latencia de auto-configuración.

Tabla 4.4: Resultados de latencias para la métrica de auto-configuración.

Latencias en ms		
Eddystone	Gatt	Ethernet
134,268	151,53	1005,785

latencias absolutas vistas desde la experiencia del usuario en obtener rápidamente la información de los sensores.

Los resultados obtenidos para los BLE TIM fueron los esperados con latencias similares, con apenas 17.26 ms de diferencia. En cambio, para un Ethernet - TIM podría esperarse una latencia menor, por su condición de conexión cableada y ser estático típicamente, pero los resultados promedios fueron superiores en 1 segundo respecto a los BLE - TIM. Existen varias causas para este resultado, las cuales vienen dada por tres factores:

- Los TIM utilizan hardware diferente. Ambos TIM usan procesadores RISC que operan a 16 MHz, pero el TIM nrf51422 [260] tiene un poco más de recursos que el Arduino Mega, ya que está basado en el ARM Cortex M0 [261] (32- bit CPU y 16kB RAM), mientras que ATmega2560 [262] se basa en un microcontrolador AVR de 8 bits con 8KB SRAM.
- Cada TIM hace uso de una plataforma de desarrollo diferente. Por un lado, en el caso de BLE TIM, ambos fueron programados con Mbed, que es una plataforma para administrar nodos IoT basados en ARM que optimiza el firmware para dispositivos con recursos de hardware de restricción. Por otro lado, Arduino es una plataforma de desarrollo que incluye una capa de abstracción que permite una programación más simple e intuitiva, incluso para aquellos que no tienen experiencia en IoT, pero a costa de optimizar los recursos administrados, lo que afecta las latencias de configuración y la cantidad de memoria de programa usada.
- El uso del Ethernet shield requiere una conexión en serie con la CPU, lo que ralentiza el descubrimiento del módulo Ethernet.

4.7.2.2. Latencia de auto-registro (Self-register) y telemetría (Telemetry)

Para obtener las latencias de Self-register and Telemetry usando los dos TIM BLE fue necesario desarrollar una aplicación Android como NCAP, donde se añadieron ciertos TAGS representativos del evento que estaba sucediendo. Con la ayuda del Log de Android Studio e identificando el TAG correspondiente podemos obtener el timestamp de cada evento.

En el caso del beacon TIM, la latencia self-register corresponde al tiempo transcurrido desde que el NCAP recibe la primera trama proveniente del Eddystone, ya sea [TLM],

[URL] o [UUID] hasta que el NCAP logre recibir toda la información de configuración del TIM, este evento es marcado con la etiqueta [MAReg]. Es importante destacar que el TIM utiliza un protocolo ligero eficiente (LP4S-six) descrito en el capítulo anterior. La figura 4.30 muestra la secuencia de ventanas de la aplicación Android que el usuario verá desde que el NCAP descubre al TIM que ha entrado en el rango de cobertura BLE del NCAP, en la primera ventana, luego aparece la segunda ventana cuando llegue cualquiera de las tramas Eddystone y la última ventana aparece solo cuando el TIM se ha logrado registrar automáticamente y mostrará dinámicamente en dicha ventana la información de los sensores y actuadores descubiertos.

La latencia self-register ha sido denotada como t1 y calculada como la diferencia de los timestamp de MAReg con las tramas Eddystone mencionadas. Cada vez que llegue un nuevo dato del sensor al NCAP, este será marcado con la etiqueta [TLM]-DATA ANALOG. El cual será usado para obtener la latencia de Telemetría.

Para el caso de un TIM Gatt server, el proceso para obtener la latencia self-register comienza cuando el usuario selecciona al TIM deseado dentro de la lista mostrada de los dispositivos BLE detectados por el NCAP. El evento de "paring"del TIM con el NCAP ha sido marcado con la etiqueta [MP], como se aprecia en la primera ventana de la figura 4.31. Una vez pareado el TIM envía a través del LP4S protocol la configuración del mote, para de esta forma auto-registrarse en el NCAP, este evento utiliza la misma etiqueta [MAReg] usada con los beacons. La segunda ventana mostrada en la figura, también se genera dinámicamente en función del hardware descubierto en el TIM. EL evento de lecturas de sensores es marcado con la etiqueta [CHR]-DATA ANALOG, utilizado para el cálculo de nuevas métrica.

La figura 4.32 muestra la secuencia de eventos y tags asociados usados para calcular la latencia self-register a nivel de cloud (t2) y las latencias de telemetría a nivel de NCAP

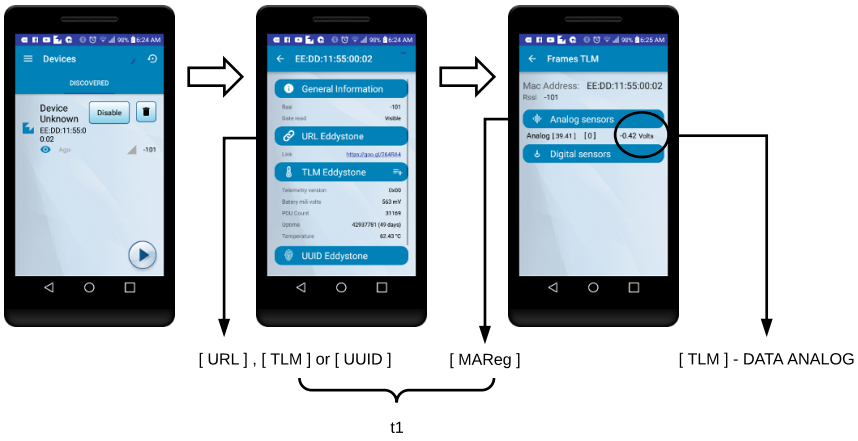


Figura 4.30: Secuencia de pantallas en la aplicación Android para el uso de beacons.

(t3) y cloud (t4) respectivamente. Estas secuencias son comunes para ambos TIM BLE (Eddystone y GATT server).

La latencia self-register a nivel de cloud (t2) representa el tiempo transcurrido desde que llega la primera trama Eddystone o el mote fue pareado hasta que la trama de configuración ha llegado a la cloud. Como el NCAP está conectado al MQTT server de la cloud y subscrito al tópico de configuración, puede saber en qué momento ha llegado dicha información a la cloud. Este evento ha sido marcado con la etiqueta [JR] y a pesar de que implica un pequeño retardo de regreso de la trama hacia el lugar de origen, nos permite garantizar un sincronismo de los tiempos sin tener que lidiar con ajustes de time zone por la ubicación y distancia entre la cloud y los nodos sensores en estos experimentos. En este tramo NCAP - Cloud el protocolo utilizado es el LP4S-Json detallado en el capítulo anterior.

Las latencias de telemetría para cualquiera de los TIM son calculadas desde que los TIM obtienen un nuevo dato de los sensores en función del esquema de muestreo empleado, hasta que dicha data llegue al NCAP (t3). Es decir, la latencia se calcula como la diferencia entre los timestamps asociados a las etiquetas [TLM]-DATA ANALOG (Eddystone) o [CHR]-DATA ANALOG con la etiqueta [PJ] que representa el evento cuando el NCAP ya dispone de la data de los sensores y procede a publicarla en el tópico de lecturas en la cloud. Para la latencia de telemetría a nivel de cloud (t4), se tomarán en cuenta los timestamps asociados a las etiquetas de DATA ANALOG con la etiqueta [JR] que aparece cuando el dato publicado llega a la cloud, en este caso el NCAP se ha subscrito al tópico de lectura de sensores en la cloud.

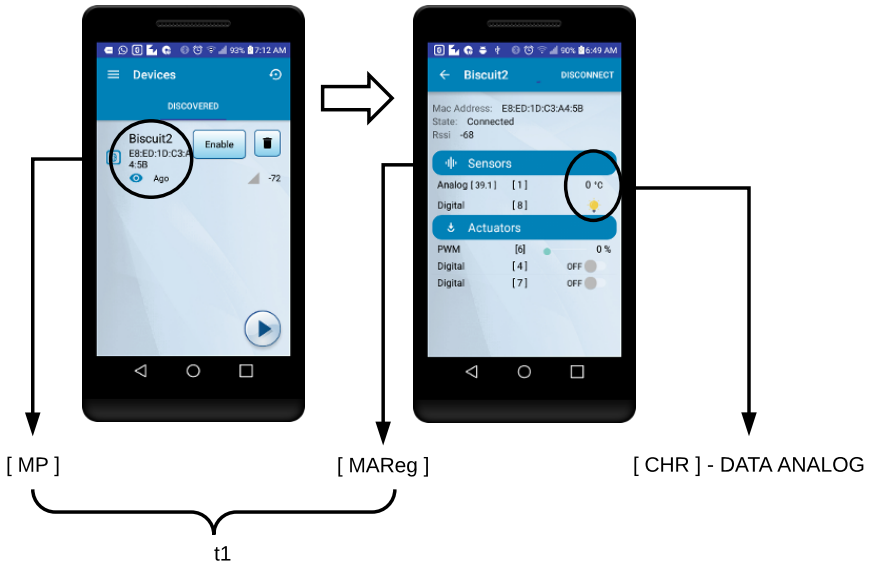


Figura 4.31: Secuencia de pantallas en la aplicación Android para el uso de GATT server.

Los timestamps en el Android Studio log son expresados en milisegundos para una mayor precisión en los cálculos de latencia. La figura 4.33 muestra un extracto de dicho log, donde se han resaltados con círculos rojos por ejemplo las etiquetas asociadas para el cálculo de la latencia self-register en un Eddystone beacon. En este caso la latencia self-register a nivel de NCAP (t1) fue de 6,411 s y la misma latencia a nivel cloud (t2) fue de 7.04 s.

Para el caso de Ethernet - TIM el procedimiento para obtener las mismas latencias difiere un poco a la conseguida con los BLE - TIM, ya que los actores (NCAP y TIM) varían en recursos y prestaciones de hardware y software. Para TIM más estáticos, como nuestro caso que presenta una conexión cableada o incluso utilizando módulos de comunicación Wifi, es fácil conseguir encaminar la información y por ende el NCAP puede ser alojado tanto en mini-computadoras al estilo de las Raspberry-Pi 3, como en máquinas virtuales dentro de servidores más potentes. Lo mismo sucede con el

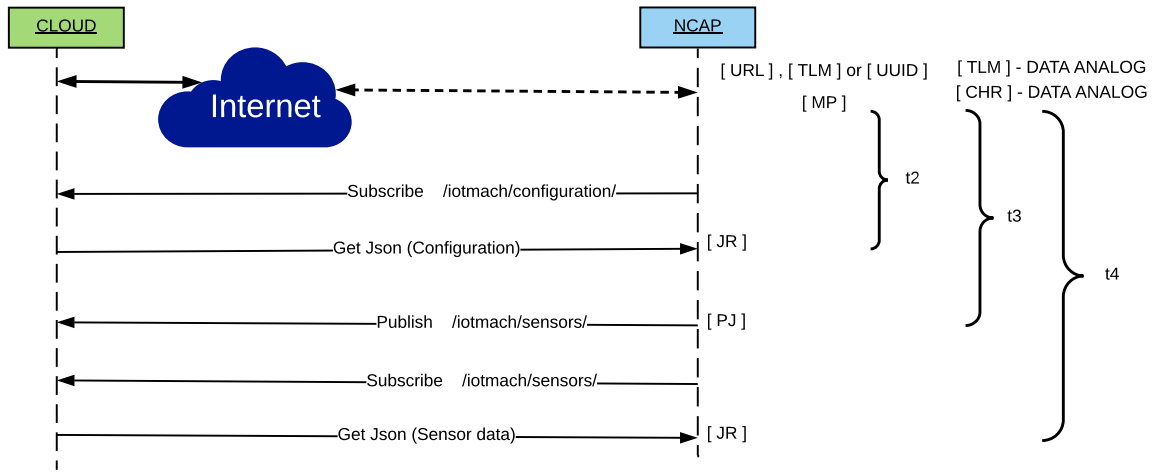


Figura 4.32: Secuencias comunes para obtener las latencias usando TIM - BLE.

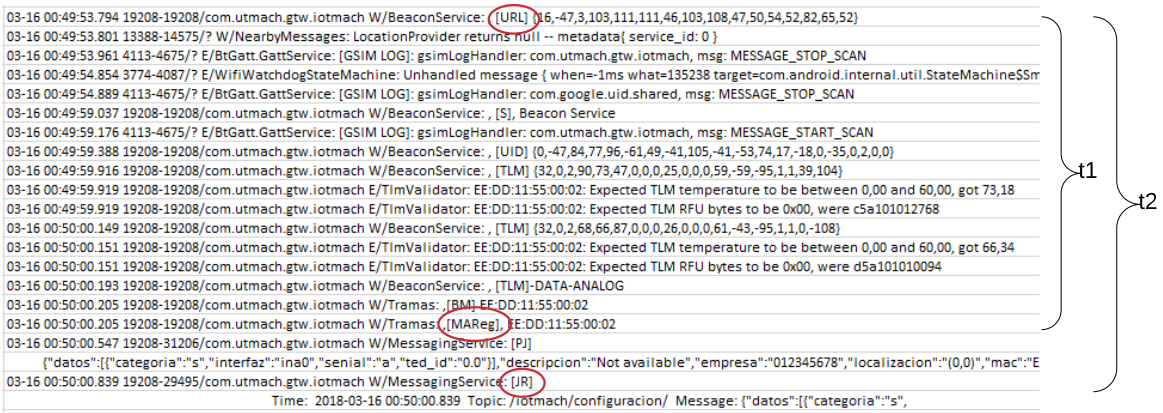


Figura 4.33: Ejemplo de un log en Android Studio para la medición de latencias.

Ethernet - TIM usado en nuestros experimentos que dispone de los recursos necesario para disponer de clientes MQTT en su firmware y conectarse directamente a servidor MQTT locales o remotos. En nuestras pruebas hemos alojado el NCAP en una máquina virtual Centos sobre la laptop antes mencionada. Este NCAP además dispone de un broker MQTT local y funciones de Fog computing, por tanto nuestro Ethernet - TIM publicará directamente tanto las tramas de configuración como de telemetría en el MQTT broker local del NCAP, el cual a su vez replicará dicha información hacia la cloud siguiendo los mismos esquemas mencionados anteriormente.

La figura 4.34 recoge las secuencias necesarias para obtener tanto las latencias de self-configuration como la de telemetría. La latencia de self-configuration a nivel de NCAP (t1) es medida desde el momento que el TIM ha reconocido su hardware y lo publica en el tópico de configuración del NCAP hasta que este logre extraer la metadata del Json recibido del protocolo LP4S-Json usado. La misma latencia, pero a nivel cloud (t2) es medida en cambio hasta que el NCAP comprueba que dicho Json ha llegado a la cloud. Esto es posible dado que en el NCAP se dispone también de un cliente MQTT suscrito al tópico de configuración de la cloud, corriendo solo con fines de depuración y para estos experimentos.

Las latencias de telemetría (t3) a nivel de NCAP y a nivel cloud (t4) son medidas desde que el TIM dispone de una nueva lectura de sus sensores hasta que el NCAP logre extraer los datos del sensor del Json para t3 o hasta que dicho dato esté disponible en la cloud en t4.

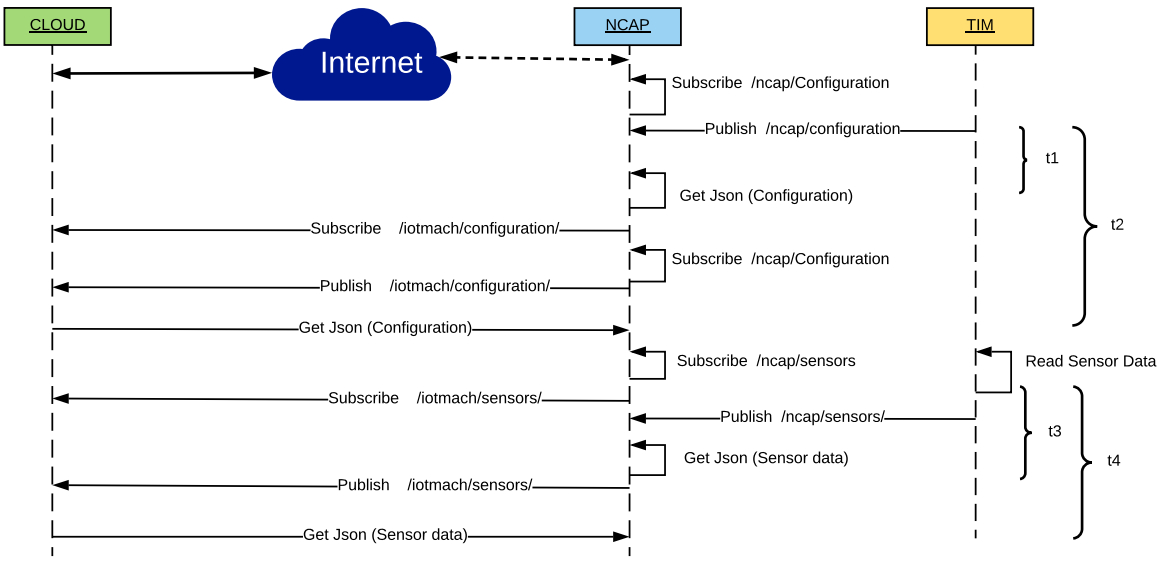


Figura 4.34: Secuencias para obtener las latencias usando Ethernet - TIM.

Tabla 4.5: Resultados de latencias para las métricas de auto-registro y temetría.

Experimentos	Latencia (Auto-registro) (ms)		Latencia (Telemetría) (ms)	
	t1 (NCAP)	t2 (Cloud)	t3 (NCAP)	t4 (Cloud)
BLE - Beacon (Adv = 250 ms)	5655,21	6557,71	1156,23	1808,62
BLE - GATT	1700,79	2541,86	1040,29	1274,71
Ethernet	739,46	1040,46	404,79	672,14

Para todas las métricas usando el Ethernet - TIM, los timestamp asociados a los eventos mencionados, fueron extraídos de la base de datos (Sqlite) local del NCAP donde fueron insertadas cada una de las tramas Json publicadas o leídas de los tópicos de configuración o lecturas. Para gestionar dicha base de datos y procesar toda la información recolectada en cada experimento se utilizó el software Navicat Premium. La figura 4.35 muestra la tabla de registros de tramas de configuración leídas y publicadas en el NCAP.

Los resultados promedios obtenidos de los 45 experimentos realizados para cada TIM para obtener las latencias de Self-register y Telemetry para BLE - TIM y Ethernet - TIM fueron consolidados en la tabla 4.5.

Los cuales muestran resultados esperados y reconocidos por la comunidad científica y trabajo propio de los autores. Por ejemplo, las latencias de self-register siempre son mayores a las de telemetría independientemente del hardware y el medio de comunicación. Con una diferencia de 4.5 s para el Eddystone beacon, 660,5 ms para el GATT server y 334,67 ms para Ethernet, todos a nivel del NCAP. Análisis similar ocurre para la diferencia entre las latencias a nivel NCAP versus la Cloud, que la primera al constituir una fase intermedia y necesaria para que la WSN alcance la cloud, los valores siempre

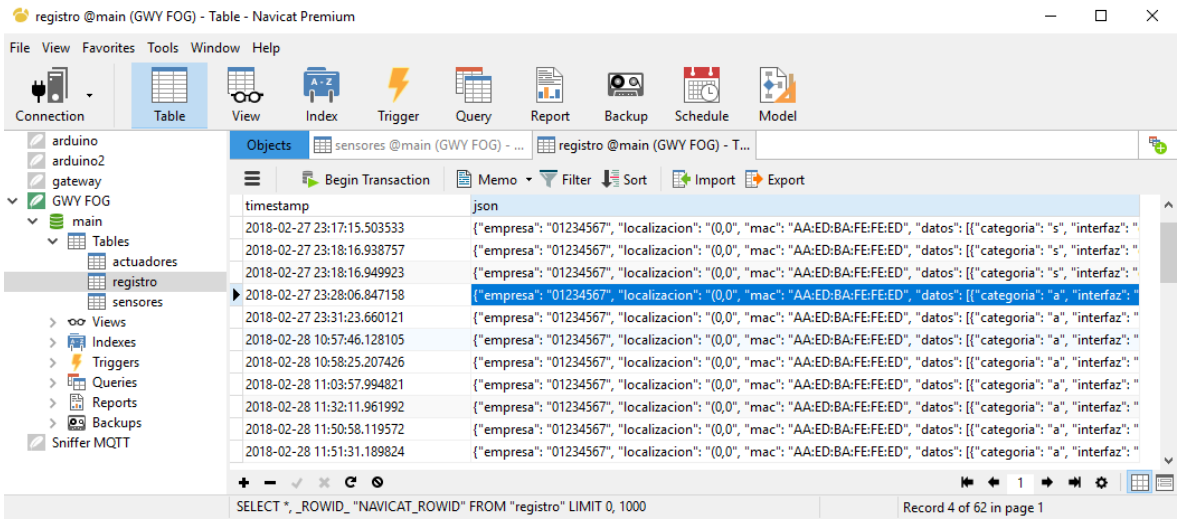


Figura 4.35: Ejemplo de tramas de configuración en la base de datos local del NCAP.

serán menores. No obstante, los resultados aquí logrados nos permiten tener valores referenciales para futuros trabajos con diferentes medios de comunicación en un ámbito IoT global real. Las diferencias para la latencia self-register entre los niveles NCAP y cloud fueron de 902 ms, 841.07 ms y 301 ms para el Eddystone, GATT y Ethernet respectivamente. Con un promedio total de solo 681.35 ms. En cambio, para la latencia de telemetría las diferencias fueron de 652.39 ms, 234.42 ms y 267.35 ms, con un promedio de apenas 384.72 ms. El promedio total de las diferencias de latencias entre ambos niveles es de medio segundo, exactamente de 533.03 ms. Otros valores referenciales a recordar será que la menor latencia, la consigue el módulo Ethernet con 404.79 ms en latencias de telemtría a nivel NCAP y la mayor latencia corresponde a Self-register con 6.5 s, conseguida por el Eddystone beacon a nivel cloud.

4.7.2.3. Latencia de Auto-calibración

Está métrica nos permite medir uno de los valores agregados de nuestra arquitectura y es medida desde la experiencia de un usuario o técnico de mantenimiento cuando ha calibrado uno de los sensores auto-registrados en el sistema IoT. De forma similar a las métricas anteriores los experimentos se realizan tanto a nivel de NCAP como de la cloud. En ambos casos el experimento fue realizado utilizando el Ethernet -TIM y los timestamps asociados fueron gestionados de la base de datos local igual que las latencias anteriores sobre el mismo Ethernet - TIM. Tanto para la latencia de Auto-calibración a nivel de NCAP (t1) como a nivel de cloud (t2) el proceso comienza cuando el usuario desde un browser de internet ingresa nuevos parámetros de calibración. La latencia en sí, será la diferencia de tiempos entre el momento que dichos parámetros son publicados en el tópico de calibración hasta que el nuevo dato del sensor ya calibrado esté disponible en el NCAP (t1) o en la cloud (t2). La figura 4.36 ilustra el proceso mencionado, destacando que ambas métricas se ven afectadas por un tiempo aleatorio entre la publicación de calibración hasta que llegue el nuevo dato del sensor proveniente del TIM. Este tiempo aleatorio depende del criterio de lectura y tiempo de muestreo definido. Sin embargo, este tiempo aleatorio tiene la misma distribución para ambas latencias sin afectar los resultados.

Los resultados promedios de una serie de 40 ensayos realizados para esta métrica aparecen en la tabla 4.6. En la cual podemos observar que la diferencia de 319.54 ms entre los niveles NCAP y Cloud se corresponden con la diferencia promedio entre estos niveles descrita en la sección anterior.

También debemos añadir que las diferencias entre NCAP y cloud de los experimentos están afectadas por las latencias aleatorias propias del proveedor de Internet en ambos extremos. Del lado de la cloud, el servidor dispone de una conexión cableada con el

Tabla 4.6: Resultados de latencias para la métrica de auto-calibración.

Experimento	Latencia (Auto-calibración) (ms)	
	t1 (NCAP)	t2 (Cloud)
Ethernet	3478	3797.54

nodo central de comunicación de la Universidad Técnica de Machala, pero no dispone de ancho de banda prioritario, sino que compite con el resto de usuarios del campus universitario. Del lado del NCAP en los experimentos, la laptop se conectó vía wifi a Tnternet como un usuario mas del campus universitario.

Comparando las latencias auto-calibración y la de self-configuration (ver tabla 4.4) en el mismo hardware Ethernet, vemos que es de 2.47 segundos y aparentemente afectada en 1 segundo por la latencia de self-configuration, pero no es así, ya que para el cálculo de esta métrica, el Ethernet - TIM ya estaba corriendo por tanto los valores de latencias de self-configuration no afectan a esta latencia, vista desde la experiencia de un usuario de obtener lo más rápido posible una medición con mayor precisión acto seguido de ingresar los nuevos parámetros de calibración. Pero, si está afectada por el tiempo aleatorio (random time) mostrado en la figura 4.36, que puede variar de 0 segundos hasta un máximo de 1 segundo, que fue el tiempo de muestreo (sample time) programado para estos experimentos.

Para efectos de estos experimentos y demostrar la funcionalidad de auto-calibración de la arquitectura propuesta, se tomó un ajuste lineal para la calibración (ca) dado por la expresión:

$$ca = fm * Data + fa \tag{4.1}$$

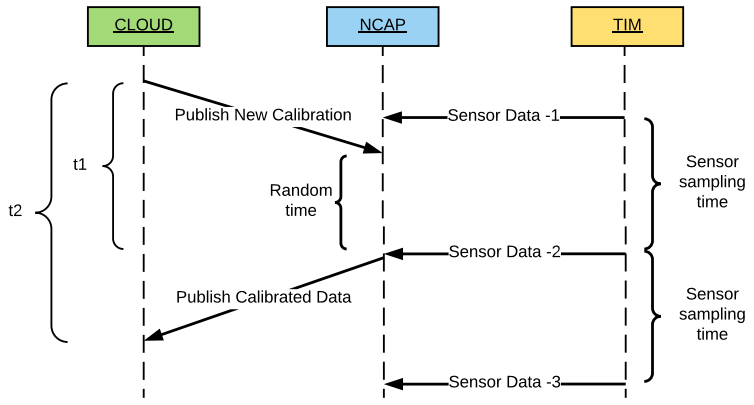


Figura 4.36: Secuencias para obtener la latencia de Auto-Calibración.

donde el usuario ingresa los parámetros de fm (factor multiplicativo) y fa (factor aditivo) a través de la interfaz web gráfica ya mostrada en la figura 4.20. No obstante, nuevas correcciones, sean estas cuadráticas, exponenciales o personalizadas pueden ser añadidas en la arquitectura en función del tipo de sensores y aplicaciones requeridas por los usuarios y empresas finales.

4.7.2.4. Latencias de auto-registro y telemetría en escenarios sin línea de vista

En aras de conocer el impacto de la latencia en el rendimiento de la infraestructura en aplicaciones reales IoT donde la distancia y la presencia de obstáculos entre el NCAP y TIM puedan comprometer incluso la conexión y la experiencia vista desde el usuario para gestionar sensores y actuadores rápidamente, se añadió un experimento más donde no exista línea de vista NoLoS (Non Light-of-Sight) entre TIM y NCAP. Para esto se utilizó como el Gatt server como TIM y un smartphone como NCAP en el escenario físico mostrado en la figura 4.37. La figura solo muestra el escenario NLoS, aunque el mismo sitio se usó para el escenario LoS (Light-of-Sight) pero en el mismo nivel sin obstáculos. Tenga en cuenta que el escenario NLoS es muy agresivo, ya que se considera atravesar un piso del edificio. Como se puede apreciar, la ubicación de ambos dispositivos ubican este experimento cercano al extremo de distancia máxima actual de 10 m para BLE en los Android smartphones actuales.

Específicamente, los experimentos se realizaron en un edificio donde tanto el NCAP como el TIM estaban en habitaciones pequeñas (un dormitorio y un comedor). Los muebles actuales estaban hechos principalmente de madera contrachapada, aunque algunos elementos contenían metal (por ejemplo, había sillas de aluminio y una base de colchón metálica). Las paredes tenían 9 cm de grosor y estaban hechas de paneles de yeso. Los pisos estaban hechos de hormigón ligero y estaban cubiertos de cerámica, con un espesor total de 15 cm. La distancia entre el transmisor y el receptor (NCAP y TIM) nunca superó los 10 m, que generalmente es el máximo admitido por el estándar de comunicación.

En este caso los experimentos permitieron obtener las latencias de auto-registro y telemetría tanto a nivel NCAP como cloud. La metodología fue exactamente la misma a la descrita anteriormente para el GATT server TIM en escenarios con línea de vista (LoS - Line of sight).

En aras de facilitar la comparación de las métricas mencionadas en ambos escenarios (LoS vs NoLoS), los resultados obtenidos anteriormente para GATT server fueron consolidados junto con el escenario NoLoS en la tabla 4.7. Para ambos casos los resultados mostrados son los promedios de la misma cantidad de ensayos en los diferentes escenarios descritos.

Tabla 4.7: Resultados de latencias para las métricas de auto-registro y temetría en escenarios NoLoS.

Experimentos	Latencia (Auto-registro) (ms)		Latencia (Telemetría) (ms)	
	t1 (NCAP)	t2 (Cloud)	t3 (NCAP)	t4 (Cloud)
BLE - GATT (LoS)	1700,79	2541,86	1040,29	1274,71
BLE - GATT (NoLoS)	3154,79	4523,21	1374,43	1618,93

Los experimentos de esta métrica arrojan como resultados una diferencia de latencias entre los niveles NACP y cloud de 1.36 s para auto-registro y 244.55 ms para telemetría. Este último incluso inferior a la media calculada para todos los TIM con LoS. Comparando los dos escenarios LoS vs NoLoS, encontramos una latencia auto-registro para NoLoS a nivel NCAP de 1.45 s superior al escenario LoS y de casi 2 segundos (1.98 s) a nivel cloud. En cambio, para las latencias de telemetría a nivel NCAP y cloud fueron similares con 334.14 ms y 344.22 ms superiores al escenario LoS.

No obstante, estos resultados para un escenario de NoLoS para BLE - TIM son excelentes, considerando que se consiguió un máximo de 4.52 s en la auto-registro ya a nivel de la cloud y un mínimo de 1.37 s de latencia de telemetría a nivel NCAP. Lo que los convierten en una excelente opción para implementar aplicaciones IoT con escenarios

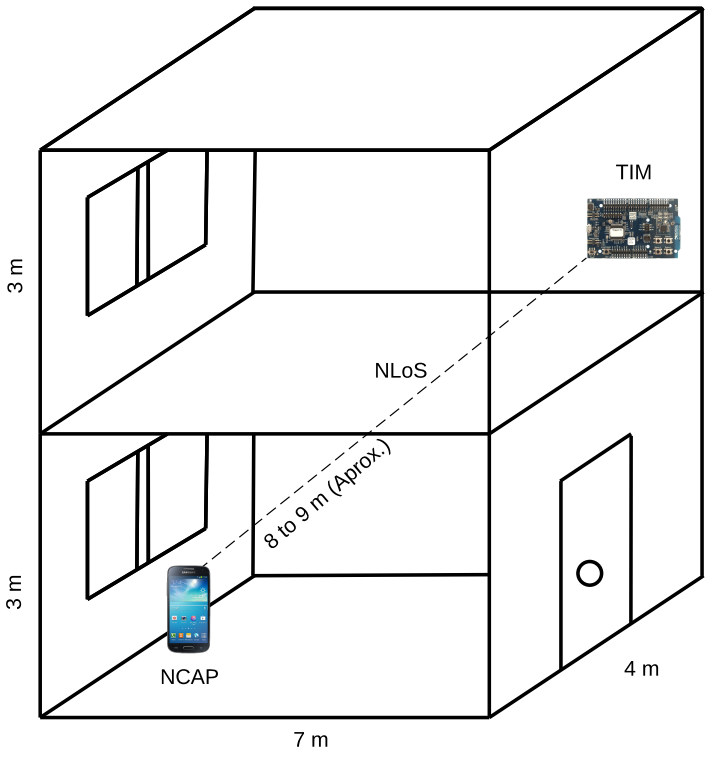


Figura 4.37: Escenario físico para los experimentos sin línea de vista (NoLoS).

de NoLoS, con obstáculos y ubicaciones al límite de la cobertura BLE. Algunos de estas aplicaciones pueden ser en Home Automation, con obstáculos de paredes y losas, muy bien demostrados en este experimento, como para Agricultura de precisión con obstáculos de follaje y plantas.

4.7.3. Análisis de resultados

En la sección anterior fue explicado con detalle cómo realizar los experimentos necesarios para obtener las métricas de latencia que permiten determinar el rendimiento de la arquitectura propuesta para que los TIM puedan ser auto-detectados tanto a nivel de la cloud en un típico ámbito IoT como a nivel del NCAP, permitiendo incluso la posibilidad de Fog computing. Conclusiones acerca de los resultados también fueron mostrados después de haber mostrado la metodología para el cálculo de cada métrica. No obstante, en está sección analizaremos con mayor profundidad el impacto de las latencias totales, vista desde la experiencia de un usuario para obtener de forma inmediata las mediciones de los sensores desde el power-on de los TIM usados. Las latencias totales o absolutas fueron calculadas desde el power-on de los TIM para ambos escenarios con LoS o NLoS con las siguientes expresiones:

$$\text{Total self-R time (NCAP o Cloud)} = \text{Self-R time} + \text{Self-C time} \tag{4.2}$$

$$\text{Total telemetry time (NCAP o Cloud)} = \text{Self-T time} + \text{Self-C time} \tag{4.3}$$

Donde:

- Self-R: auto-registro
- Self-C: auto-configuración
- Self-T: auto-telemetría

La figura 4.38 muestra una comparación de tales latencias totales para un escenario LoS para los tres TIM utilizados en los experimentos (*t1* a *t4* son los tiempos definidos previamente en la figura 4.34). Como se puede observar, las latencias totales de autorregistro (t1 y t2 calculadas por la ecuación 4.2) de los TIM de GATT y Ethernet son similares, siendo esta última menor en solo 107.08 ms en la capa de NCAP.

Los hallazgos más interesantes están relacionados con las latencias de telemetría y se destacan en la figura 4.38 con círculos rojos. Para la latencia de telemetría total (t3 calculada por la ecuación 4.3 en la capa NCAP ambos TIM BLE muestran los datos del sensor más rápido que la versión Ethernet (Eddystone es 120.08 ms más rápido y

GATT TIM, 218.75 ms más rápido). En la capa cloud (t4 calculada por la ecuación 4.3), el GATT TIM es 251.68 ms más rápido que el TIM de Ethernet, a pesar de que los experimentos se llevaron a cabo utilizando una cloud remota en Internet. Por lo tanto, se puede concluir que la solución de telemetría más rápida corresponde a BLE GATT TIM, tanto en NCAP como en capas de cloud, cuando el usuario solicita datos desde un teléfono inteligente en el mismo lugar donde están los sensores o cuando los recibe a través de un control remoto navegador web.

Finalmente, la figura 4.39 compara las latencias totales del TIM del GATT en los escenarios LoS / NLoS. Las latencias totales para Ethernet con LoS también se han incluido.

Una vez más, la solución con el GATT TIM demuestra ser una buena solución para los escenarios NLoS IoT: como se puede observar a la vista de los valores dentro de los círculos azules, hay una diferencia de solo 115.39 ms en la capa NCAP y solo 92.54 ms en la capa de la nube.

Otra conclusión interesante es que las soluciones basadas en Mobile Opportunistic Fog (Mobo-Fog), no solo demuestran su factibilidad de uso en aplicaciones IoT reales, sino su rapidez y flexibilidad, permitiendo el uso tanto de dispositivos BLE clásicos (GATT server) como Eddystone beacons en aplicaciones de telemetría. Mobile Opportunistic Fog (Mobo-Fog), es un término bautizado por el autor, para aquellos NCAP implementados en dispositivos móviles con la capacidad suficiente para llegar a una

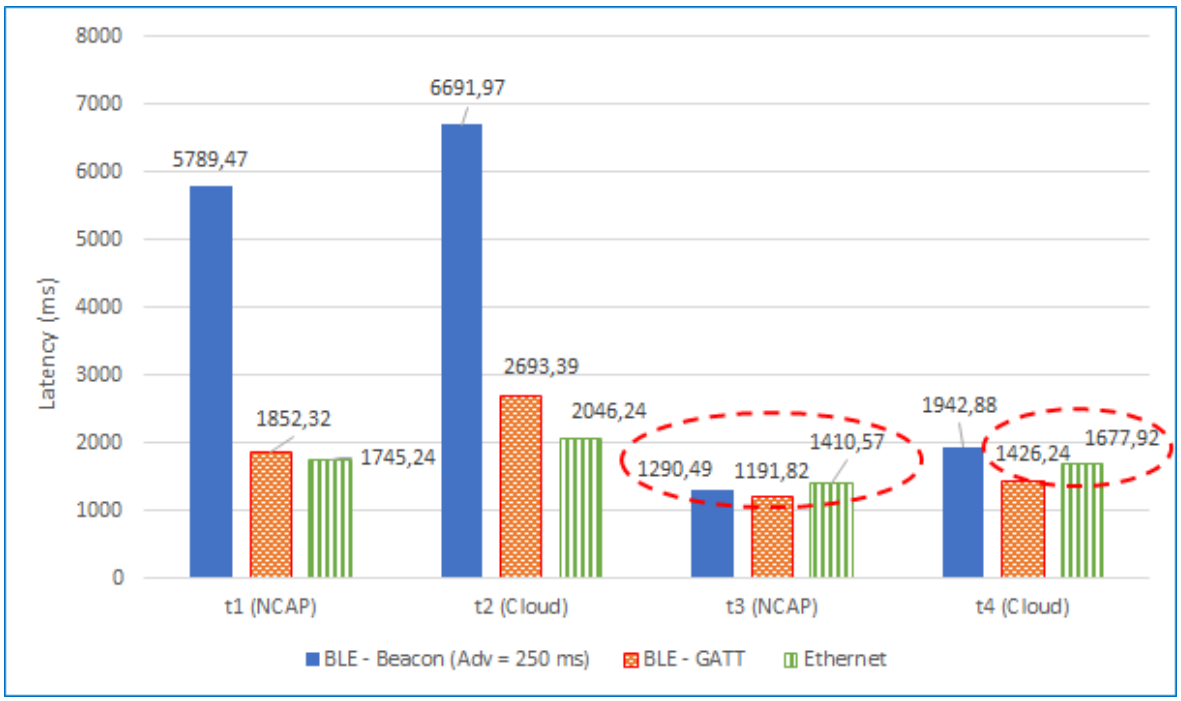


Figura 4.38: Latencias totales de auto-registro y telemetría con línea de vista (LoS).

WSN y oportunistamente capturar la data de los sensores in situ, permitiendo la toma de decisiones al usuario en tiempo real. Los resultados aquí analizados, que muestran a una Mobo-Fob más rápida y dinámica que las Classic Fog (basadas en conexiones estáticas como el caso de Ethernet o Wi-fi) demuestran que estas serán cada vez más usadas en aplicaciones IoT.

4.8. Conclusiones

Como contribución a la evolución del IoT hacia la WoT, este capítulo presentó una arquitectura plug and play genérica, flexible, dinámica y rápida basada en Virtual TEDs que permiten la interoperabilidad de distintos tipos de motes (TIM), en cuanto a hardware y medios de comunicación. Y estos con el uso protocolos ligeros como el LP4S logran auto-describirse, auto-registrarse, auto-configurarse y auto-calibrarse, de forma rápida y eficiente. Los TED virtuales son guardados en bases de datos tanto en el servidor como en los NCAP permitiendo realizar Fog computing a nivel de la WSN. También se presentó como parte de la arquitectura propuesta, una aplicación web de monitoreo, residente en la cloud que permite entre otras funciones la gestión de sensores

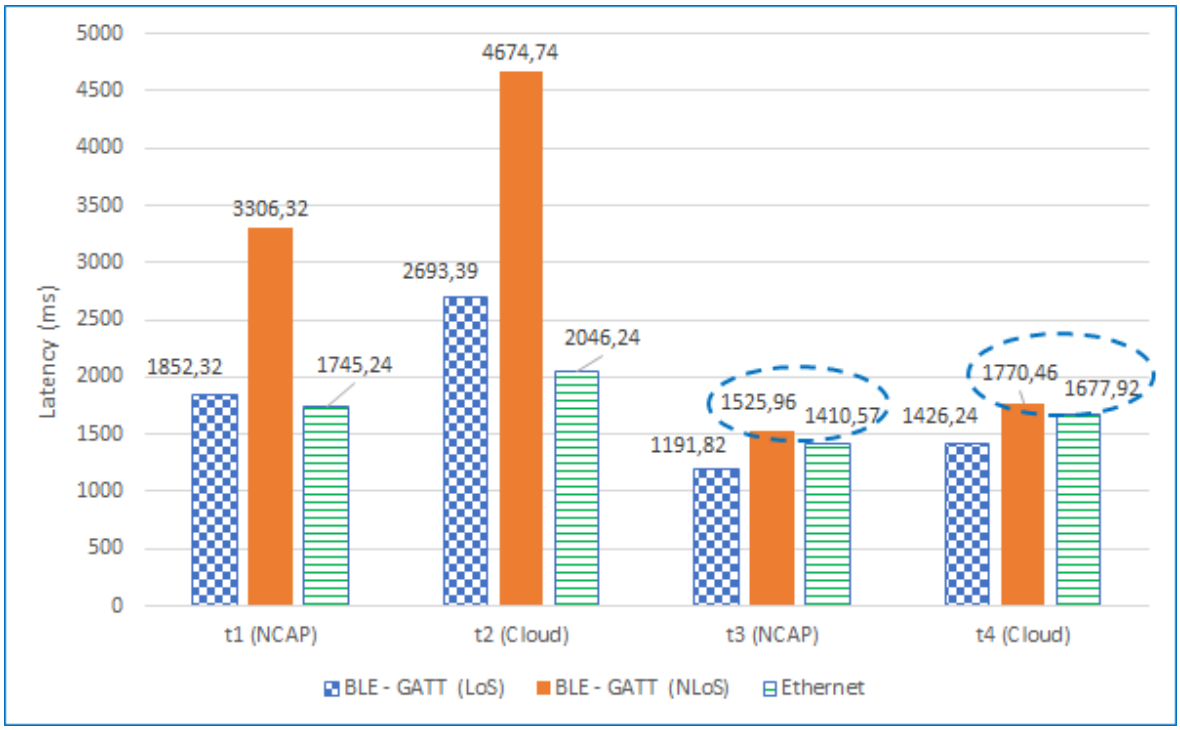


Figura 4.39: Latencias totales de auto-registro y telemetría para ambos escenarios LoS y NoLoS.

y actuadores de forma remota, generando dashboards dinámicos auto-configurables y la actualización de los parámetros de calibración de los TEDS en tiempo real.

Para determinar el rendimiento del sistema propuesto, se evaluaron tres TIMs: dos beacons BLE y un sensor inteligente basado en Ethernet en diferentes escenarios de LoS y NLoS. Las pruebas midieron diversas latencias con el objetivo de evaluar objetivamente la experiencia del usuario. Los resultados obtenidos muestran que el sistema es realmente rápido para la mayoría de los dominios de aplicación, pudiendo auto-registrarse y auto-configurarse en menos de 3 s sensores inteligentes colocados en España incluso cuando la nube estaba ubicada en Ecuador. En escenarios similares, el sistema puede mostrar datos del sensor a usuarios remotos en menos de 2 s, lo que indica que un usuario puede acceder rápidamente a la información del transductor.

Los resultados demostraron la superioridad integral del GATT server en términos de latencias totales, vistas como la experiencia del usuario en obtener datos del sensor lo más rápido posible desde que el TIM es energizado (powered-on). Se definió el término de Mobile Opportunistic Fog y su tendencia de uso sobre los Classic Fog. Por último la arquitectura propuesta permite el uso de tecnologías emergentes como los Bluetooth Low Energy Beacons y Physical web.

Capítulo 5

Trabajos Futuros y en Curso

5.1. Trabajos futuros según tendencias tecnológicas

En el capítulo 2 se analizó el estado del arte de los sistemas de telemetría y los principales actores en el ámbito del Internet de las cosas. Muchos de los cuales marcan la tendencia tecnológica actual y por ende el camino de los proyectos de investigación mas cercanos. Por tanto, una continuación de nuestra investigación se verá complementada con las siguientes tendencias:

- Redes de largo alcance: Los principales actores serán Lora, Sigfox y LTE. En nuestra opinión, Lora tendrá mayor impacto en los sistemas de telemetría de bajo coste y con requerimientos de bajo consumo de potencia. Es por ello, que los esfuerzos serán encaminados a construir redes tipo Lora en nuestro entorno, para garantizar la ubicuidad tan deseada.
- Redes de Corto alcance: Las redes tipo mesh dominarán este sector, siendo Bluetooth Mesh y Thread los que mayor presencia tendrán en este mercado. Por ejemplo, Bluetooth mesh permitirá una fácil actualización del software en dispositivos BLE desde la versión BLE 4.0. Además BLE mesh ya forma parte de las especificaciones de Bluetooth Low Energy lo cual garantiza su permanencia al menos una década más en el mercado. La baja potencia de este tipo de red y posibilidad de implementación en hardware restrictivos es otro plus, que junto a la seguridad añadida con encriptación de mensajes de extremo a extremo serán sin duda nuestra elección en proyectos futuros.
- Hardware: Los dispositivos multi-protocolo serán la tendencia marcada en el futuro. Nordic Semiconductor, seguirá siendo nuestra elección mas cercana con su nRF52840 que integra Bluetooth Low Energy 5.0, Thread, 802.15.4 y ANT en un mismo SoC [263].

- Integración de redes: Los fabricantes optarán no solo por el uso de dispositivos SoC multi-protocolos, sino además la integración de múltiples dispositivos y tecnologías embebidas en un producto OEM final. Fipy [95], es un ejemplo clásico que integra las 5 redes mas importantes del presente y futuro mas cercano. Sin duda nuestros próximos proyectos estarán basados en Fipy.

- Sistemas operativos y plataformas de desarrollo: En este sector muchos actores seguirán compitiendo por el mismo nicho de mercado sin avisorar un claro ganador. No obstante los lenguajes de programación en C, Node y Python serán los mas usados por desarrolladores haciendo uso de plataformas como, m-bed [46], node-red [45] y Pymark [264]. Pycom ha liberado su Pymark plugin y podrá ser integrado en los populares editores de código Atom [265], Sublime [266], Visual Studio Code [267] y PyCharm [268].

Típicamente en trabajos de investigación, la sección de trabajos futuros es incluida junto con las conclusiones del trabajo realizado. En este caso, hemos decidido ubicarlos en un capítulo aparte, ya que al término de la redacción de este documento fue lanzado un nuevo proyecto de investigación IoT. En este proyecto, se ha hecho uso de los protocolos ligeros LP4S explicados en el capítulo 3, la arquitectura basada en VTED mostrada en el capítulo 4, el sistema telemetría multipropósito propuesto en el capítulo 2 y algunas de las tendencias tecnológicas antes mencionadas. En dicho proyecto, se han logrado ya algunos resultados parciales relevantes. A continuación, se exponen los principales hitos conseguidos.

5.2. Proyecto de control de riego de una producción de banano basado en IoT.

En esta sección daremos a conocer el alcance del proyecto denominado Control de riesgo automatizado de una parcela de banano basado en el Internet de las Cosas, aprobado con la Resolución No.396/2016 [269] por el Consejo Universitario de la Universidad Técnica de Machala. Teniendo como objetivo general, la de implementar un sistema de riego automatizado en la plantación de banano en la Granja Experimental Santa Inés de la UTMACHALA, utilizando Internet de las Cosas para lograr un uso eficiente del agua, manteniendo la productividad y un fruto sano.

Algunos de los objetivos específicos definidos fueron:

- Realizar el levantamiento planimétrico de la granja con la plantación de banano objeto de estudio.

- Seleccionar los elementos del sistema de automatización: actuadores (Electroválvulas), sensores (Humedad, presión, etc.) y módulos de comunicación.
- Diseñar el sistema de telemetría y definir la ubicación de los elementos del sistema en el plano de la plantación objeto de estudio.
- Diseñar y construir el sistema automatizado, software de control y monitoreo para el riego del banano.
- Realizar pruebas de laboratorio.
- Implementar los componentes eléctricos, electrónicos y comunicación del sistema de riego.
- Instalar, configurar y validar el sistema final.

Este proyecto aún esta en ejecución y a continuación será mostrado el diagrama general diseñado, ubicación real de los componentes en la finca Santa Inés e instalación de electroválvulas y panel de control.

5.2.1. Diagrama general del sistema de telemetría del proyecto.

La figura 5.1, muestra el diagrama general del sistema de telemetría, el cual fue diseñado en función de alguno de nuestros aportes como son la arquitectura de de telemetría multipropósito basada en WSN, motes con protocolos ligeros embebidos que garanticen, bajo consumo de potencia, baja latencia y calidad de servicio, sistema basado en TEDS virtuales para el reconocimiento plug-and-play de sensores y actuadores y el uso de dashboards dinámicos. En este caso, el dominio de aplicación fue el de agricultura de precisión y específicamente en cultivo de banano.

Para realizar el diseño del sistema de control mostrado en la figura 5.1, se comenzó con un diagnóstico del sistema de riego actual en la finca Santa Inés, ubicada en los predios de la Unidad académica de Ciencias Agropecuarias de la Universidad Técnica de Machala. Un aspecto a destacar es que dicha finca experimental se encuentra en producción, es decir, el banano que se produce es exportado. El sistema de riego está formado básicamente por una bomba de agua, un pozo artesanal y cinco zonas de riego con un respectivo plan de riego que permite regar máximo dos zonas a la vez, en aras, de mantener la presión necesaria en el extremo de las plumas de regadío,de acuerdo al caudal de agua disponible con las tuberías instaladas existentes.

El control del paso de agua, aún se lo realiza de forma manual, es decir, existe una llave de agua por cada zona de riego y dos llaves maestras que divide en dos grandes secciones a la finca mencionada. En una sección se encuentran las zonas de riego 1 y 2 y en la

otra las zonas de riego 3, 4 y 5. En nuestro diseño hemos concebido una electroválvula por cada llave manual existente, para lograr una automatización efectiva del riego. En la figura 5.1 estas zonas de riego están representadas por recuadros sombreados y es perfectamente reconocible la trayectoria del agua, desde el pozo, sacada e impulsada por la bomba controlando las zonas a ser regadas por medio de cinco electroválvulas; las cuales serán accionadas de forma manual, remota o automática a través de un Panel de control, del cual hablaremos mas adelante.

El control automático del riego, es decir la decisión de apertura o cierre de las electroválvulas indicadas, se lo realiza a partir de la información de los sensores del sistema IoT diseñado. Los sensores seleccionados fueron, sensores de humedad y temperatura del suelo, sensor de presión de la bomba de agua, sensor de humedad y temperatura ambiental. Es importante destacar que el diseño admite el ingreso de mas variables, sean éstas provenientes de sensores en tiempo real, ingresadas manualmente o tomadas de bases de datos disponibles, respecto al clima de la región donde este sistema sea implementado.

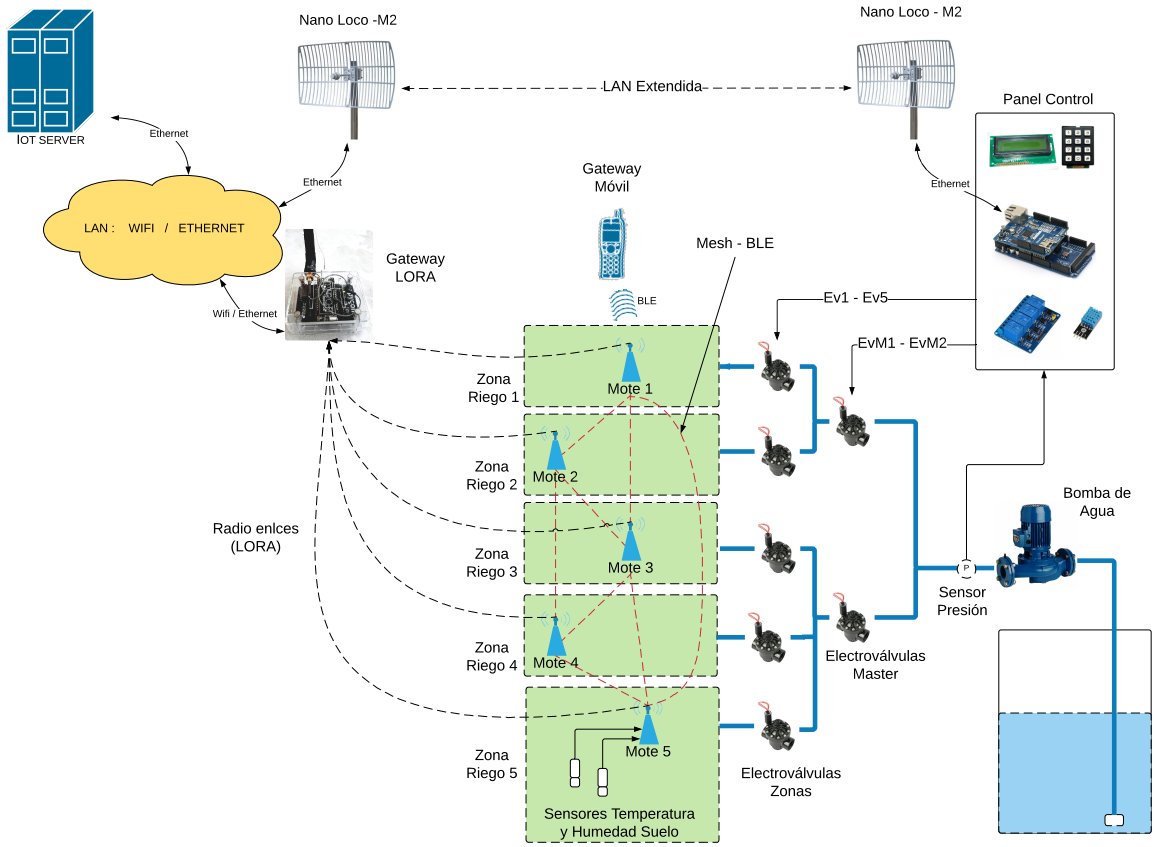


Figura 5.1: Diagrama general sistema de control de riego basado en IoT.

Los sensores de presión y estado climático son leídos directamente por el panel de control, en cambio en cada zona de riego fue necesario colocar un mote con dos sensores de humedad y temperatura del suelo. En la figura 5.1, solo se muestran dichos sensores acoplados al mote 5 ubicado en la zona de riego 5 como ejemplo representativo de las cinco de zonas de riego. Estos sensores deben ser colocados alrededor de la zona radicular de la planta de banano, es decir uno casi en la superficie y el otro al finalizar las raíces de la planta, con el fin de garantizar un riego óptimo, utilizando el agua justa y necesaria para conseguir un fruto de calidad. En la figura 5.2 podemos apreciar una imagen de la ubicación de ambos sensores en la tierra respecto a las raíces de la planta de banano.

Como motes o transductores inteligentes del sistema fue seleccionado al módulo Lopy4 [270] de Pycom que tiene integrada cuatro redes, LoRa, Sigfox, WiFi y Bluetooth, de las cuales haremos uso de Lora para conectarse directamente a la cloud IotMach por medio de un respectivo Gateway Lora y a su vez crear una mesh BLE redundante, como se aprecia en la figura 5.1. La mesh BLE permitirá extraer localmente la información en

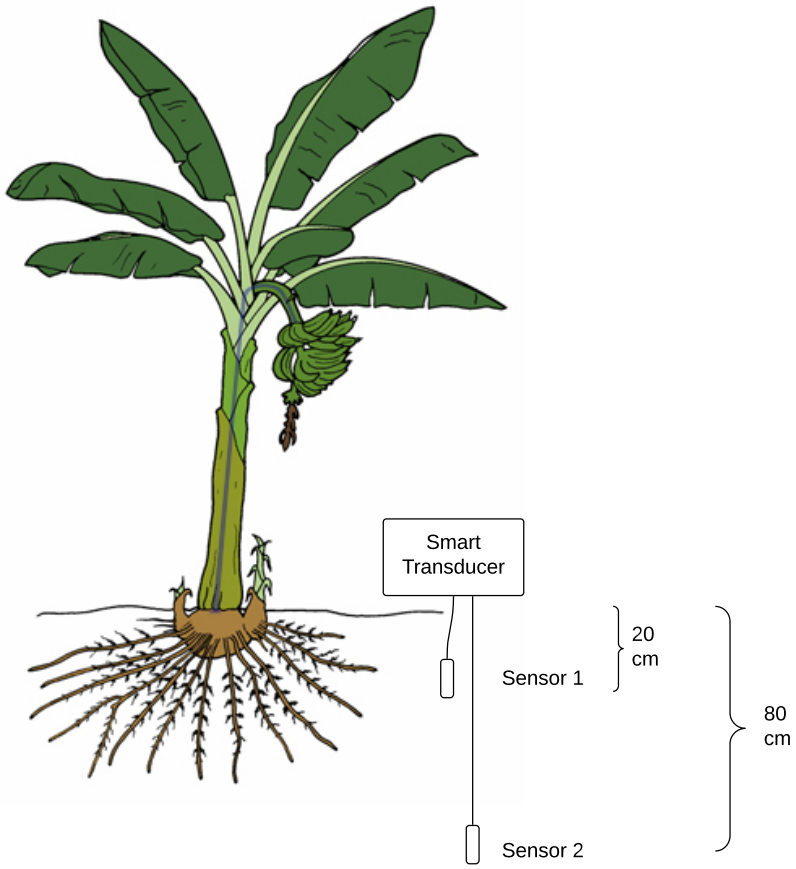


Figura 5.2: Ubicación de sensores de humedad y temperatura del suelo respecto a la zona radicular.

tiempo real de los sensores así como la apertura y cierre de electroválvulas por medio del Gateway Móvil basado en teléfonos inteligentes Android, explicado en el capítulo 2. Consiguiendo así flexibilidad y redundancia en el sistema de telemetría, que derivarán en estudios posteriores comparativos de la eficiencia de ambas redes en este dominio de aplicación.

En nuestro diseño fue necesario incluir un panel de control, para conseguir la aceptación de la automatización por parte del agricultor, acercándolo a una gestión centralizada de la apertura y cierre de las electroválvulas de forma manual. El panel de control tiene un diseño robusto industrial y dispone de un conjunto de botoneras, donde se puede controlar el riego especifico de cada zona, así como las llaves máster. Además de las botoneras y asociadas a la experiencia del agricultor se añadió un display que permita visualizar in situ el valor de las variables disponibles en el panel. Estas variables son la presión de la bomba de agua y la temperatura y humedad ambiental. El cerebro del panel de control está formado por un Arduino Mega con un shield Ethernet instalado. Con la ayuda de un banco de relés las salidas digitales del Arduino permitirán el control de cada electroválvula. Para conectar dicho panel a la cloud IoT se optó por implementar una LAN extendida utilizando radio enlaces Nano Loco - M2, conectadas a la red IP del servidor por un lado y al módulo Ethernet shield insertado en el Arduino Mega, al otro extremo. Esta LAN extendida no solo permitirá la conexión del panel de control sino la posibilidad de nuevos gateways de comunicación de corto alcance que se implementen en el futuro.

En la próxima sección serán mostrados los resultados parciales que ser han podido conseguir dentro de la ejecución del citado proyecto. Básicamente, serán mostradas las instalaciones de algunos de los elementos adquiridos para la implementación del sistema de control de riego explicado anteriormente.

5.3. Resultados parciales

Dando cumplimiento a uno de los objetivos específicos del proyecto se definió la ubicación de los elementos del sistema en el plano de la plantación objeto de estudio. La figura 5.3 muestra una vista superior de la plantación de banano de la finca Santa Inés donde se esta ejecutando el proyecto de control de riego automatizado.

Para una percepción real del sistema se utilizó una imagen actualizada de Google Maps de la ubicación de la finca en la ciudad de Machala, provincia de El Oro, Ecuador. Sobre dicho mapa 3D, se muestran las zonas reales de riego, ubicación de los motes por zona y la presencia de una caseta de control de bomba ubicada al lado del pozo de agua y en su interior se encuentra la bomba propiamente dicha y el panel de control. Como se puede observar en dicha imagen, las zonas de riego se encuentran separadas por una

carretera de acceso a la Unidad de ciencias agropecuarias, motivo por el cual las llaves máster separan en dos secciones las zonas de riego, en el lado derecho de la carretera se encuentras las zonas 1 y 2 y del lado izquierdo las zonas 3, 4 y 5 respectivamente. También existe otra carretera que divide la zona cultivada con el campus universitario, donde en uno de sus edificios se encuentra instalado el servidor IoTMach. Encima de la caseta de la bomba y del edificio donde se encuentra el servidor fueron instaladas las antenas para crear la LAN extendida. El servidor adquirido se muestra en la figura 5.4.

En la figura 5.5 a) se muestra la parte exterior de la caseta de control de bombeo, donde se destaca en su techo la antena del Nanoloco-M2 con tecnología Airmax como uno de los extremos de la LAN extendida incluida en el diseño. En la figura 5.5 b), se puede observar el interior de dicha caseta donde ha sido instalado el panel de control

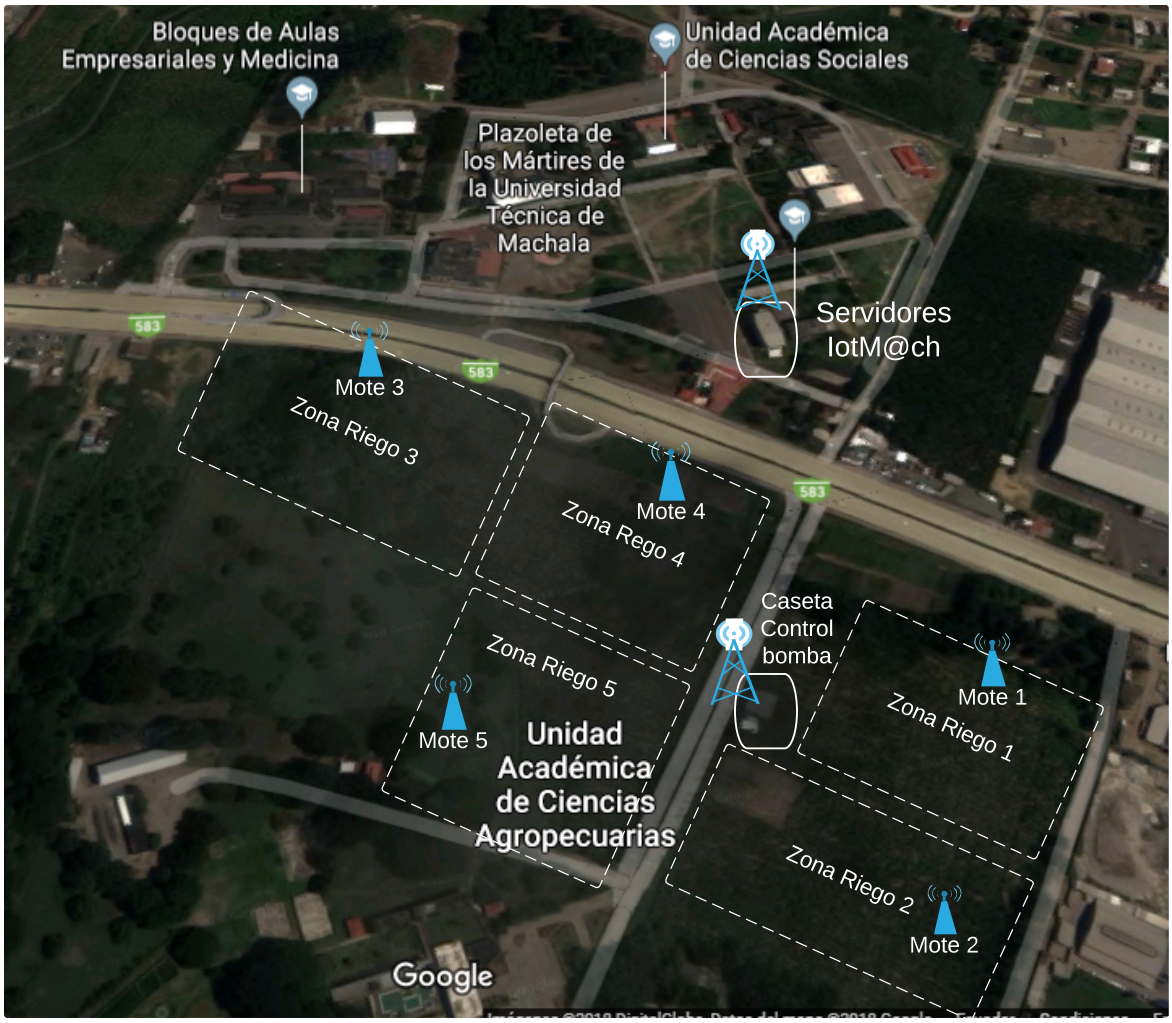


Figura 5.3: Vista superior de Google Maps de la finca Santa Ines.

junto a la bomba de agua y en la figura 5.5 c) se puede apreciar el sistema de conexiones internas del panel de control.

Por último, se muestra como resultado parcial la instalación de uno de los actuadores más importantes de un sistema de riego como son las electroválvulas, las cuales fueron instaladas en paralelo con las llaves manuales ya existentes en la finca. En la figura 5.6

Figura 5.4: Servidor donde se aloja la cloud IotMach.

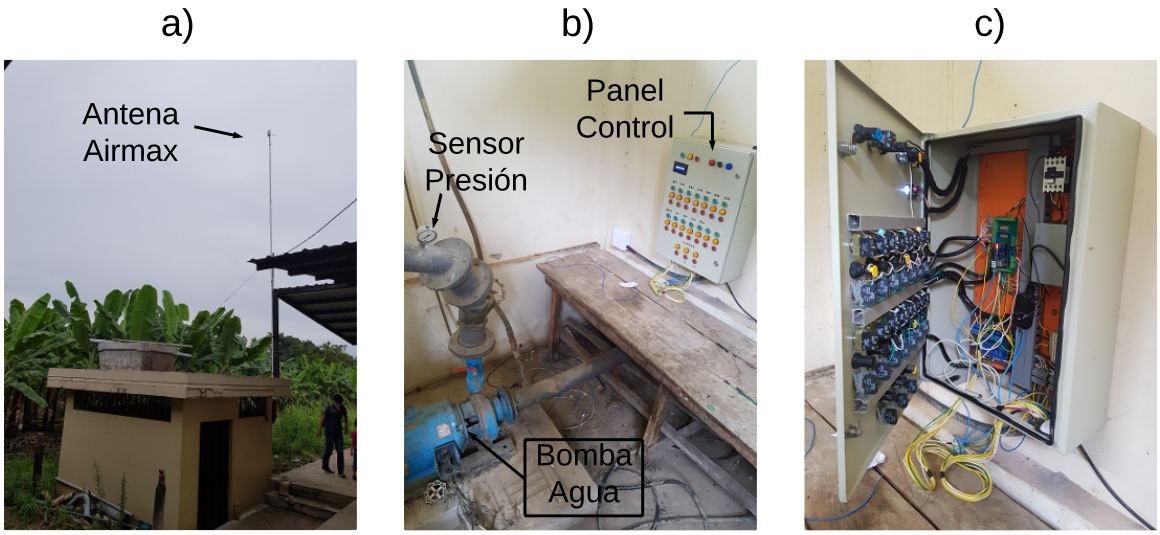


Figura 5.5: Caseta de control de la bomba de agua.

Figura 5.6: Instalación de una electroválvula en campo.

a) se muestran el instante cuando la tubería principal fue cortada para poder insertar una T y lograr tener en paralelos ambas opciones, manual o controlada. En la figura 5.6 b) se muestra la conexión en paralelo de dicha electroválvula antes de ser protegida con paredes de hormigón como se aprecia en la figura 5.6 c).

Capítulo 6

Conclusiones

El objetivo principal de esta tesis es el diseño y desarrollo de un sistema de telemetría multipropósito que obtengan variables de un proceso en tiempo real y las guarde en una Cloud Computing para el Internet de las Cosas. La red de sensores inalámbricos formada por motes deben ser programados con protocolos ligeros que permitan su integración al sistema de telemetría de forma plug & play. La arquitectura IoT que se proponga debe hacer uso de estándares de comunicación y descripción del mote que facilite la interconexión con todo tipo de redes a nivel IP de forma óptima en cuanto a latencia, consumo de energía y calidad de servicio.

6.1. Conclusiones de cada capítulo

Esta sección presenta las principales conclusiones del trabajo e investigaciones derivadas de cada capítulo. Los cuales a su vez, están basados en las publicaciones científicas logradas como producto de nuestra investigación.

6.1.1. Sistema de telemetría

En este capítulo se propuso un sistema de telemetría multi-propósito IoT teniendo en cuenta la heterogeneidad actual de las tecnologías involucradas en los dominios de sensores, red y aplicaciones. Dentro de la arquitectura general propuesta se propone la interconexión de diferentes tipos de redes, corto y largo alcance, motes aislados y LAN extendida por medio del protocolo MQTT.

Diferentes escenarios de acceso a la cloud son analizados, exponiendo la flexibilidad y escalabilidad de dicha arquitectura. Por último se muestra un ejemplo de implementación de un testbed IoT, desarrollando firmware para motes BLE y un gateway basado en teléfono inteligente con sistema operativo Android.

Previamente, se analizó el concepto de telemetría, su presencia en los sistemas SCADA y DSC, así como la evolución de estos hasta llegar a conceptos actuales de Industria 4.0, IIoT e IoE. Luego, se revisó el estado del arte de cada dominio, comenzando por sensores, actuadores, transductores inteligentes y tecnologías de comunicación, mostrando ejemplos comerciales representativos de cada uno, para que sirva como guía o punto de partida en la implementación de sistemas de telemetría IoT actuales. El estado del arte del gateway y diferentes modelos de despliegue de cloud computing aquí mostrados, ofrecen también una guía rápida de elección de plataformas IoT actuales disponibles en el mercado.

6.1.2. Protocolos ligeros

En este capítulo se presentó la familia de protocolos ligeros LP4S (Lightweight Protocol for Sensors), la cual constituye uno de los principales aportes de este trabajo. LP4S fue diseñado para ser usado en aplicaciones IoT fundamentalmente, por tanto, este es capaz de proporcionar comunicaciones de baja latencia y bajo consumo de energía. Además, permite la auto-detección y auto-registro de un mote a través de mecanismos plug-and-play, para sistemas de telemetría IoT. La familia LP4S está formada por los protocolos LP4S-6 (Lightweight Protocol for Sensors-Six), LP4S-X (Lightweight Protocol for Sensors-Extended) y LP4S-J (Lightweight Protocol for Sensors-JSON). Este último dispone a su vez de los subprotocolos LP4S-JC (Lightweight Protocol for Sensors-Json Configuration), LP4S-JR (Lightweight Protocol for Sensors-Json Read) y LP4S-JW (Lightweight Protocol for Sensors-Json Write), que añaden robustez a la gestión de sensores y actuadores.

En dependencia de los recursos del procesador del mote usado, se debe utilizar el protocolo mas adecuado. Por ejemplo, LP4S-J, permite describir al mote haciendo uso de mensajes del tipo JSON, los cuales pueden ser perfectamente en motes con cierta capacidad de procesamiento y memoria. En cambio, cuando se dispone de hardware restrictivo de estos recursos se recomienda el uso del protocolo LP4S-6, el cual permite describir en solo 6 bytes al mote, haciendo uso de un eficiente algoritmo. Para motes ubicados en un punto intermedio de los ya mencionados, se recomienda al protocolo LP4S-X, que es una extensión del LP4S-6, facilitando la programación del firmware del mote y legibilidad de la mensajería.

Para determinar qué tipo de conectividad es la mejor opción para ser utilizado en diferentes contextos y aplicaciones, se llevaron a cabo varios experimentos. Donde se demostró la viabilidad de uso del protocolo LP4S-6, embebido en un beacon Bluetooth Low Energy comercial Eddystone. Comparativamente con los beacons basados en GATT server, los resultados demostraron la superioridad de la solución propuesta con LP4S-6

en términos de consumo de energía y latencia cuando varios usuarios están conectados a un mote. Por otra parte, los resultados también sugirieron el uso del protocolo LP4S-6 para escenarios en los que la latencia no era una restricción, pero el bajo consumo de energía era esencial.

Finalmente, es importante mencionar que, aunque la familia protocolo fue desarrollada y probada dentro de un protocolo restrictivo como los beacons Eddystone, también puede implementarse en otros dispositivos IoT similares, proporcionando comunicaciones eficientes y rápidas.

6.1.3. TEDS virtuales

Como contribución a la evolución de IoT hacia WoT, este capítulo presentó una arquitectura plug-and-play genérica, flexible, dinámica y rápida basada en IEEE 21451 VTED, que permite la interoperabilidad de diferentes tipos de transductores inteligentes en términos de hardware y interfaces de comunicación. Estos VTEDS, gracias al uso de protocolos livianos como LP4S, son capaces de auto-describirse, auto-registrarse, auto-configurarse y calibrar automáticamente transductores inteligentes de forma rápida y eficiente.

Este capítulo aporta con el diseño de una arquitectura basada en VTEDs y una herramienta de monitoreo web intuitiva, que es capaz de administrar sensores y actuadores de forma remota, generar dashboards dinámicos auto-configurables o actualizar los parámetros de calibración de los VTEDS en tiempo real. Para determinar el rendimiento del sistema propuesto, se evaluó en diferentes escenarios con línea de vista (LoS -Light of Sigh) y sin línea de vista (NLoS - Non Light of Sigh) al hacer uso de tres TIMs (Transducer Interface Module): dos beacons BLE y un sensor inteligente basado en Ethernet. Las pruebas midieron diversas latencias con el objetivo de evaluar objetivamente la experiencia del usuario. Los resultados obtenidos muestran que el sistema es realmente rápido para la mayoría de los dominios de aplicación, pudiendo auto-registrarse y auto-configurarse en menos de 3 segundos sensores inteligentes colocados en España incluso cuando la cloud estaba ubicada en Ecuador. En escenarios similares, el sistema puede mostrar datos del sensor a usuarios remotos en menos de 2 segundos, lo que indica que un usuario puede acceder rápidamente a la información del transductor.

6.1.4. Trabajos futuros

En este capítulo se presentaron las principales tendencias tecnológicas en el campo de la telemetría basada en IoT, que estarán en la mira de nuestros próximos proyectos y de la comunidad científica en general. Las tendencias mostradas están enfocadas en tecnologías ya disponibles para redes de largo alcance, redes de corto alcance, hardware

de los transductores inteligentes basados en dispositivos multi-protocolos con integración de múltiples redes de comunicación, sistemas operativos, lenguajes de programación y plataformas de desarrollo para firmwares de los motes.

Por último, fueron mostrados tanto el diseño como los avances parciales de un proyecto de investigación aún en ejecución, dirigido al dominio de aplicación de agricultura de precisión y específicamente en el control de riego de una producción de banano en una granja experimental. El diagrama general del sistema de control de riego mostrado aquí, hace uso de los principales aportes científicos de esta tesis, como son la creación de redes WSN, basados en dispositivos multi-protocolos, multi-redes, mesh-Bluetooth Low Energy, LAN extendida, arquitectura plug-and-play con VTEDS y uso de la familia de protocolos ligeros LP4S en diferentes dominios IoT. Además se mostraron la ubicación real de los componentes mencionados en una imagen 3D de Google maps, instalaciones de electroválvulas y panel de control en campo. Un panel de control fue introducido para conseguir entre otros motivos una transición aceptable por parte del personal agrícola de un control de riego automatizado.

6.2. Producción científica

Como resultado del trabajo realizado durante el desarrollo de esta tesis, se han conseguido diferentes hitos de producción científica que se detallan a continuación.

6.2.1. Participación en Proyectos de Investigación

La investigación realizada para esta tesis ha contribuido a los siguientes proyectos:

- Proyectos culminados:
  - Implementación de un gateway genérico en dispositivos móviles Android para monitoreo y control de una WSN. 2017. (RES-390/2017) [271]
  - Sistema de telemetría multipropósito basado en una red de sensores inalámbricos para el Internet de las cosas (IoT). 2017.(RES-607/2016). [272]
- Proyectos en curso:
  - Smart Campus. (RES-362/2017) [273].
  - Control de riego automatizado de una parcela de banano con el Internet de las cosas. (RES-396/2016) [269].

6.2.2. Publicaciones

Los contenidos de la tesis han sido publicados en las siguientes revistas especializadas y foros.

6.2.2.1. Revistas JCR

1. Dixys L. Hernández Rojas, Tiago M Fernández-Caramés, Paula Fraga-Lamas and Carlos J Escudero. Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing trough BLE beacons in IoT telemetry applications. *Sensors.* **2017**, *18*, 1. Impact factor 2016: 2.677 (Q1/T1 10/58 INSTRUMENTS & INSTRUMENTATION).

2. Dixys L. Hernández Rojas, Tiago M Fernández-Caramés, Paula Fraga-Lamas and Carlos J Escudero. A Plug-and-Play Human-Centered Virtual TEDS Architecture for the Web of Things. *Sensors.* **2018**, *18*, 2052. Impact factor 2016: 2.677 (Q1/T1 10/58 INSTRUMENTS & INSTRUMENTATION).

3. Jordi Duran-Pesantez, Dixys Hernández-Rojas, Bertha Mazon-Olivo, Carlos J. Escudero, Ivan Ramirez-Morales and Patricio Quizhpe-Cordero. Detection of feeding behavior in fish, using a low-cost hydroacoustic signal processing-based system. To be published in *Aquaculture.* **2018**. Impact Factor 2016: 2.57 (Q1)

4. Bertha Mazon-Olivo, Alberto Pan Bermudez, Dixys Hernández-Rojas and Hernan Maza-Salinas Rules Engine and Complex Event Processor in the Context of Internet of Things for Precision Agriculture.To be published in *Computers and Electronics in Agriculture.* **2018**. Impact Factor 2016: 2.201 (Q1)

6.2.2.2. Revistas SJR

1. Dixys Hernandez-Rojas, Bertha Mazon-Olivo, Jhonny Novillo-Vicuña, Gustavo Belduma-Vacacela and Carlos J Escudero. Iot Android gateway for monitoring and control a WSN. In *Communications in Computer and Information Science, Proceedings of the Third International Conference on Technologies Trends*, Springer 2018; Volume 798, pp. 18-32. Impact factor 2016: 0.162 (Q3/T2 COMPUTER SCIENCE (MISCELLANEOUS)).

6.2.2.3. Congresos Internacionales

1. Dixys Hernandez-Rojas. Impacto del Internet de las Cosas en los mercados. Publicado en las memorias del II Congreso Internacional de Economía: El fomento de la competencia, innovación y mercado. Machala, Ecuador, 28-30 Octubre 2015.

2. Dixys Hernandez-Rojas. Internet de las cosas: Estado del arte. Publicado en las memorias del II Congreso Internacional de Ciencia y Tecnología UTMACH. Machala, Ecuador, 23-25 Noviembre 2016.

3. Dixys Hernandez-Rojas, Bertha Mazon-Olivo, Jhonny Novillo-Vicuña, Jessenia Zaruma-Villón. Repositorio Digital Temático para grupos de I+D+i en el ámbito de las TICs. Publicado en las memorias del II Congreso Internacional de Ciencia y Tecnología UTMACH. Machala, Ecuador, 23-25 Noviembre 2016.

4. Jhonny Novillo-Vicuña, Dixys Hernandez-Rojas, Bertha Mazon-Olivo. Tablero de control para la automatización del riego basado en el Internet de las cosas. Publicado en las memorias del II Congreso Internacional de Ciencia y Tecnología UTMACH. Machala, Ecuador, 23-25 Noviembre 2016.

5. Bertha Mazon-Olivo, Dixys Hernandez-Rojas, Jhonny Novillo-Vicuña, Juan Morocho-Ajila, Karina Lovato-Loayza. Iotmach-Cloud: la Cloud Computing Open Source para el Internet de las Cosas de la Universidad Técnica de Machala. Publicado en las memorias del II Congreso Internacional de Ciencia y Tecnología UTMACH. Machala, Ecuador, 23-25 Noviembre 2016.

6. Jhonny Novillo-Vicuña, Dixys Hernandez-Rojas, Bertha Mazon-Olivo, Kevin Correa. Monitoreo en tiempo real de señales eléctricas de voltaje 110/220v a través de Arduino Uno. Publicado en las memorias del 1er Congreso Internacional de Tecnologías para el Desarrollo (TECDES). Machala, Ecuador, 20-22 Septiembre 2017.

6.2.2.4. Congresos Nacionales

1. Dixys Hernandez-Rojas, Hixya Villegas. Implementación de un sistema de monitoreo remoto de una red de impresoras multi-funcionales basado en SNMP y programado con Labview. Congreso Investigación, Universidad y Conocimiento, Loja-Ecuador, 19–20 Octubre 2014. Publicado en Revista Tecnológica-ESPOL, ISSN 1390-3659, 2015, Volumen 28, Issue 4.

2. Dixys Hernandez-Rojas, Bertha Mazon-Olivo and Ariel Campoverde. Cloud computing con herramientas open-source para Internet de las cosas. 3er Congreso Ecuatoriano de Tecnologías de la Información y la Comunicación TIC EC, Cuenca-Ecuador, 1-4 Diciembre 2015. Publicado en Revista Maskana, ISSN: 2477-8893, 2015, Volumen 6, pp.173-182

3. Dixys Hernandez-Rojas. Estado actual del Internet de las cosas. Conferencia magistral en ciclo de Conferencias de Ingeniería de Sistemas con motivo del Cua-

dragésimo tercer aniversario de la creación de la Unidad Académica de Ingeniería Civil. Machala, Ecuador, 13 Septiembre 2016.

6.2.2.5. Capítulos de Libro

1. Dixys Hernandez-Rojas, Bertha Mazon-Olivo and Carlos J Escudero. Introducción al Internet de las Cosas (IoT). *Análisis de Datos Agropecuarios*, 1st ed.; Editorial UTMACHALA, Ecuador, 2018. ISBN: 978-9942-24-120-7. [23]

Acrónimos

6LoWPAN IPV6 over Low Power Personal Area Network. 28

ADC Analog Digital Converter. 17

AMQP Advanced Message Queuing. 20, 53, 102

AP Agricultura de Precisión. 34

API Application Programming Interface. 69

BLE Bluetooth Low Enegy. 4, 49, 103

BR Bit Rate. 54

CAGR Compound Annual Growth Rate. 50

CAT-M1 Category M1. 29

CoAP Constrained Application Protocol. 3, 53

CPD Centro de Procesamiento de Datos. 2, 34

CPU Central Processing Unit. 43

DIY Do it yourself. 86

DSC Distributed control Systems. 11

EDR Enhanced Data Rate. 54

EID Ephemeral Identifier. 56

FPGA Field-Programable Gate. 53

GATT General Attribute Profile. 7

GIS Geographic Information System. 34

GPS Global Positioning System. 34

HF High Fecuency. 54

I2C Inter-Integrated Circuit. 88

IEEE Institution of Electrical and Electronic Incorporated Engineers. 7, 87

IIoT Industrial Internet of Things. 12

IoE Internet of Everything. 12

IoT Internet of things. 1

IOTMACH Internet of Things of UTMACHALA. 26

IPV4 Internet Protocol version 4. 23

IPV6 Internet Protocol version 6. 23

ISO International Organization for Standardization. 87

ITU International Telecommunication Union. 91

JMS Java Messaging Service. 102

JSON Java Script Object Notation. 65

LAN Local Area Network. 5

LDR Light Dependant Resistor. 13, 15

LDVT Linear Variable Differential Transformer. 13

LED Light Emiter Diode. 36

LF Low Frecuency. 54

Lora Long Range. 4, 20

Lora-WAN Long Range Wide Area Network. 5

LP4S Lightweight Protocol for Sensors. 8, 51

LP4S-6 Lightweight Protocol for Sensors-Six. 7, 8

LP4S-J Lightweight Protocol for Sensors-JSON. 8

LP4S-X Lightweight Protocol for Sensors-Extended. 8

LTE Long-Term Evolution. 20, 22

LWM2M Lightweight M2M. 53

LWS Leaf Wetness Sensor. 15

MAC Medium Access Control. 52

MAN Metropolitan Area Network. 5

MIT Massachusetts Institute of Technology. 1

MQTT Message Queuing Telemetry Transport. 3, 53, 102

NB-IoT NarrowBand IoT. 29

NCAP Network Capable Application Processor. 88

NFC Near-Field Communication. 20

NIS Network Information Service. 87

OGC Open Geospatial Consortium. 87

OMA Open Mobile Alliance. 53

PAN Personal Area Network. 22

PC Personal Computer. 2

PnP Plug-And-Play. 9, 86

PPK Power Profiler Kit. 73

PWM Pulse-Width Modulation. 36, 65

QoS Quality of Service. 31, 86

RDF Resource Description Framework. 98

REST Representational State Transfer. 3, 53

RFID Radio-frequency identification. 2, 20

RNC Radio Network Controller. 22

RPC Remote Procedure Call. 102

RTD Resistance Temperature Detectors. 13, 35

RTOS Real Time Operating System. 86

RTSAL Real-Time Segmentation and Labeling. 92

SBC Session Bordere Controller. 69

SCADA Supervisory Control and Data Acquisition. 11

SoC System on a chip. 68

SPI Serial Peripheral Interface. 88

SSL Secure Socket Layer. 57

SWE Sensor Web Enablement. 87

TADS Transducer Algorithm Data Sheet. 92

TEDS Transducer Electronic Data Sheet. 7, 88

TIC Tecnologia Informacion y las Comunicaciones. 34

TIM Transducer Interface Module. 88

TLM Telemetry frame. 57, 69

TML Transducer Markup Language. 95

UART Universal Asynchronous Receiver-Transmitter. 38, 88

UCLA University of California, Los Angeles. 1

UHF Ultra High Fecuency. 54

UID Unique Identifier. 56

URL Uniform resource locator. 57

USN Ubiquitous Sensor Network. 95

VTEDS Virtual Transducer Electronic Data Sheet. 7, 86

WAN Wide Area Network. 5

Wi-fi Wireless Fidelity. 5

WiMax Worldwide Interoperability for Microwave Access. 5

WoT Web of Things. 9, 49, 86

WSN Wireless Sensor Network. 4, 49

WTM Web Thing Model. 53

XML Extensible Markup Language. 91

XMPP Extensible Messaging and Presence. 3, 20, 53, 102

Bibliografía

[1] Ariel M Campoverde, Dixys L Hernández, and Bertha E Mazón. Cloud computing con herramientas open-source para internet de las cosas. *Maskana*, 6(Supl.):173–182, 2015.

[2] Janggwan Im, Seonghoon Kim, and Daeyoung Kim. Iot mashup as a service: cloud-based mashup service for the internet of things. In *Services Computing (SCC), 2013 IEEE International Conference on*, pages 462–469. IEEE, 2013.

[3] George Suciu, Victor Suciu, Alexandru Martian, Razvan Craciunescu, Alexandru Vulpe, Ioana Marcu, Simona Halunga, and Octavian Fratu. Big data, internet of things and cloud convergence–an architecture for secure e-health applications. *Journal of medical systems*, 39(11):141, 2015.

[4] George Suciu, Simona Halunga, Alexandru Vulpe, and Victor Suciu. Generic platform for iot and cloud computing interoperability study. In *Signals, Circuits and Systems (ISSCS), 2013 International Symposium on*, pages 1–4. IEEE, 2013.

[5] Chengen Wang, Zhuming Bi, and Li Da Xu. Iot and cloud computing in automation of assembly modeling systems. *IEEE Transactions on Industrial Informatics*, 10(2):1426–1434, 2014.

[6] Muhammad Intizar Ali, Naomi Ono, Mahedi Kaysar, Zia Ush Shamszaman, Thu-Le Pham, Feng Gao, Keith Griffin, and Alessandra Mileo. Real-time data analytics and event detection for iot-enabled communication systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 42:19–37, 2017.

[7] Ralf Gitzel, Simone Turring, and Sylvia Maczey. A data quality dashboard for reliability data. In *Business Informatics (CBI), 2015 IEEE 17th Conference on*, volume 1, pages 90–97. IEEE, 2015.

[8] Aimad Karkouch, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications*, 73:57–81, 2016.

[9] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[10] Stefan Mijovic, Erion Shehu, and Chiara Buratti. Comparing application layer protocols for the internet of things via experimentation. In *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*, pages 1–5. IEEE, 2016.

[11] Hongming Cai, Boyi Xu, Lihong Jiang, and Athanasios V Vasilakos. Iot-based big data storage systems in cloud computing: perspectives and challenges. *IEEE Internet of Things Journal*, 4(1):75–87, 2017.

[12] Karthik Kambatla, Giorgos Kollias, Vipin Kumar, and Ananth Grama. Trends in big data analytics. *Journal of Parallel and Distributed Computing*, 74(7):2561–2573, 2014.

[13] ABM Moniruzzaman and Syed Akhter Hossain. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*, 2013.

[14] Sjaak Wolfert, Lan Ge, Cor Verdouw, and Marc-Jeroen Bogaardt. Big data in smart farming–a review. *Agricultural Systems*, 153:69–80, 2017.

[15] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[16] Faisal Karim Shaikh, Sherali Zeadally, and Ernesto Exposito. Enabling technologies for green internet of things. *IEEE Systems Journal*, 11(2):983–994, 2017.

[17] Ovidiu Vermesan and Peter Friess. *Building the hyperconnected society: Internet of things research and innovation value chains, ecosystems and markets*, volume 43. River Publishers, 2015.

[18] Mohammad Aazam and Eui-Nam Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, pages 687–694. IEEE, 2015.

[19] Yuan Ai, Mugen Peng, and Kecheng Zhang. Edge cloud computing technologies for internet of things: A primer. *Digital Communications and Networks*, 2017.

[20] Saad Khan, Simon Parkinson, and Yongrui Qin. Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing*, 6(1):19, 2017.

[21] Universidad Técnica de Machala. Convenio Utmachala para cooperación Interinstitucional 2014. https://www.utmachala.edu.ec/archivos/portalweb/CONVENIOS%20SUSCRITOS%20EN%202014.pdf, 2014. (Accessed on 07/27/2018).

[22] Universidad Técnica de Machala. Resolución No. 342/2014 H. Consejo Universitario UTMACHALA. https://www.utmachala.edu.ec/archivos/siutmach/documentos/resoluciones/2014/Octubre%2014/334-345,347-353,355-359,361-362,364.pdf, 2014. (Accessed on 07/27/2018).

[23] Hernandez-Rojas Dixys et al. Internet de las cosas. In *Análisis de Datos Agropecuarios*, pages 72–100. Editorial UTMACH, 2018. http://repositorio.utmachala.edu.ec/handle/48000/12540 (Accessed on 07/12/2018).

[24] Dixys Hernandez-Rojas, Bertha Mazon-Olivo, Johnny Novillo-Vicuña, Carlos Escudero-Cascon, Alberto Pan-Bermudez, and Gustavo Belduma-Vacacela. Iot android gateway for monitoring and control a wsn. In *International Conference on Technology Trends*, pages 18–32. Springer, 2017.

[25] National Research Council et al. *Expanding the vision of sensor materials*. National Academies Press, 1995.

[26] C.W. de Silva. *Sensors and Actuators: Engineering System Instrumentation, Second Edition*. CRC Press, 2015.

[27] Dixys Hernández Rojas and Sergio Rodríguez Arias. Amplificador de rango dinámico programable con auto-diagnóstico en tiempo real. Cuba, CU 22573 A1, Aug 1999.

[28] Arduino - home. https://www.arduino.cc/. (Accessed on 05/01/2018).

[29] Arduino ethernet shield 2. https://store.arduino.cc/usa/arduino-ethernet-shield-2. (Accessed on 05/01/2018).

[30] Raspberry pi 3 model b - raspberry pi. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. (Accessed on 05/02/2018).

[31] Iot analytics - thingspeak internet of things. https://thingspeak.com/. (Accessed on 05/02/2018).

[32] Espressif systems - wi-fi and bluetooth chipsets and solutions. https://www.espressif.com/en. (Accessed on 05/02/2018).

[33] cloudbit™ – littlebits. https://shop.littlebits.com/products/cloudbit. (Accessed on 05/02/2018).

[34] Particle — photon family of wi-fi development kits and connectivity modules. https://www.particle.io/products/hardware/photon-wifi/. (Accessed on 05/02/2018).

[35] Beagleboard.org - black. https://beagleboard.org/black. (Accessed on 05/02/2018).

[36] Pinoccio — crowd supply. https://www.crowdsupply.com/pinoccio/mesh-sensor-network. (Accessed on 05/02/2018).

[37] All-you-need arm cortex a9 sbc development board — udoo. https://www.udoo.org/udoo-dual-and-quad/. (Accessed on 05/02/2018).

[38] Modules — samsung artik iot platform. https://www.artik.io/modules/. (Accessed on 05/02/2018).

[39] nrf52 preview dk / bluetooth low energy / products / home - ultra low power wireless solutions from nordic semiconductor. https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-Preview-DK. (Accessed on 05/02/2018).

[40] Espruino wifi. http://www.espruino.com/WiFi. (Accessed on 05/02/2018).

[41] Libelium - connecting sensors to the cloud. http://www.libelium.com/. (Accessed on 05/02/2018).

[42] Micro:bit educational foundation — micro:bit. http://microbit.org/. (Accessed on 05/02/2018).

[43] Redbear. https://redbear.cc/. (Accessed on 05/02/2018).

[44] Hexiwear — iot and wearables development platform. https://www.hexiwear.com/. (Accessed on 05/02/2018).

[45] Node-red. https://nodered.org/. (Accessed on 05/02/2018).

[46] Mbed iot platform — mbed. https://www.mbed.com/en/platform/. (Accessed on 04/30/2018).

[47] Android things — android developers. https://developer.android.com/things/. (Accessed on 05/02/2018).

[48] Contiki: The open source operating system for the internet of things. http://www.contiki-os.org/. (Accessed on 04/30/2018).

[49] Riot - the friendly operating system for the internet of things. http://www.riot-os.org/. (Accessed on 05/02/2018).

[50] Home - zephyr project. https://www.zephyrproject.org/. (Accessed on 05/02/2018).

[51] Apache mynewt. https://mynewt.apache.org/. (Accessed on 05/02/2018).

[52] Weave — nest. https://nest.com/weave/. (Accessed on 05/02/2018).

[53] fuchsia - git at google. https://fuchsia.googlesource.com/fuchsia/. (Accessed on 05/02/2018).

[54] Fuchsia · github. https://github.com/fuchsia-mirror. (Accessed on 05/02/2018).

[55] Mahdi Amiri-Kordestani and Hadj Bourdoucen. A survey on embedded open source system software for the internet of things. In *Free and Open Source Software Conference*, 2017.

[56] Home page — lora alliance™. https://lora-alliance.org/. (Accessed on 05/02/2018).

[57] S. Raza, P. Misra, Z. He, and T. Voigt. Building the internet of things with bluetooth smart. *Ad Hoc Networks*, 57:19–31, 2017.

[58] Pascal Thubert, Alexander Pelov, and Suresh Krishnan. Low-power wide-area networks at the ietf. *IEEE Communications Standards Magazine*, 1(1):76–79, 2017.

[59] Sigfox - the global communications service provider for the internet of things (iot). https://www.sigfox.com/en. (Accessed on 05/02/2018).

[60] Wi-fi halow — wi-fi alliance. https://www.wi-fi.org/discover-wi-fi/wi-fi-halow. (Accessed on 05/02/2018).

[61] Specifications — bluetooth technology website. https://www.bluetooth.com/specifications. (Accessed on 05/02/2018).

[62] Ant / ant+ defined - this is ant. https://www.thisisant.com/developer/ant-plus/ant-antplus-defined/. (Accessed on 05/02/2018).

[63] Home. https://www.threadgroup.org/. (Accessed on 05/02/2018).

[64] Zigbee alliance. http://www.zigbee.org/. (Accessed on 05/02/2018).

[65] Z-wave — safer, smarter homes start with z-wave. http://www.z-wave.com/. (Accessed on 05/02/2018).

[66] Knx association - knx association [official website]. https://www.knx.org/knx-en/index.php. (Accessed on 05/02/2018).

[67] Near field communication: What is near field communication? http://nearfieldcommunication.org/. (Accessed on 05/02/2018).

[68] Ieee council on rfid — bridging the physical to the digital. http://www.ieee-rfid.org/. (Accessed on 05/02/2018).

[69] Coap — constrained application protocol — overview. http://coap.technology/. (Accessed on 05/02/2018).

[70] Mqtt. http://mqtt.org/. (Accessed on 05/02/2018).

[71] Xmpp — xmpp main. https://xmpp.org/. (Accessed on 05/02/2018).

[72] Home — amqp. https://www.amqp.org/. (Accessed on 05/02/2018).

[73] Webrtc home — webrtc. https://webrtc.org/. (Accessed on 05/02/2018).

[74] Meshlium xtreme - the internet of things iot gateway - smartphone detection — libelium. http://www.libelium.com/products/meshlium/. (Accessed on 05/02/2018).

[75] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. Nist cloud computing standards roadmap. *NIST Special Publication*, 35:6–11, 2011.

[76] Daniele Pizzolli, Giuseppe Cossu, Daniele Santoro, Luca Capra, Corentin Dupont, Dukas Charalampos, Francesco De Pellegrini, Fabio Antonelli, and Silvio Cretti. Cloud4iot: a heterogeneous, distributed and autonomic cloud platform for the iot. In *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*, pages 476–479. IEEE, 2016.

[77] Barrie Sosinsky. *Cloud computing bible*, volume 762. John Wiley & Sons, 2010.

[78] Open source software for creating private and public clouds. https://www.openstack.org/. (Accessed on 05/02/2018).

[79] Apache cloudstack: Open source cloud computing. https://cloudstack.apache.org/. (Accessed on 05/02/2018).

[80] Ibm cloud. https://www.ibm.com/cloud/. (Accessed on 05/02/2018).

[81] Aws — cloud computing - servicios de informática en la nube. https://aws.amazon.com/es/. (Accessed on 05/02/2018).

[82] Microsoft azure: plataforma y servicios de informática en la nube. https://azure.microsoft.com/es-es/. (Accessed on 05/02/2018).

[83] Cloud computing, servicios de alojamiento y apis de google cloud — google cloud. https://cloud.google.com/. (Accessed on 05/02/2018).

[84] The things network. https://www.thethingsnetwork.org/. (Accessed on 05/02/2018).

[85] Kaa iot product development platform — iot application enablement. https://www.kaaproject.org/. (Accessed on 05/02/2018).

[86] Iot cloud platform — samsung artik cloud services. https://artik.cloud/. (Accessed on 05/02/2018).

[87] Temboo. https://temboo.com/. (Accessed on 05/02/2018).

[88] Iot platform - ibm watson iot. https://www.ibm.com/internet-of-things/spotlight/watson-iot-platform. (Accessed on 05/02/2018).

[89] Iot platform for connected devices – xively. https://www.xively.com/. (Accessed on 05/02/2018).

[90] Kiran Jot Singh and Divneet Singh Kapoor. Create your own internet of things: A survey of iot platforms. *IEEE Consumer Electronics Magazine*, 6(2):57–68, 2017.

[91] Iot cloud platform landscape — 2018 vendor list. https://www.postscapes.com/internet-of-things-platforms/. (Accessed on 05/02/2018).

[92] Iotmach. http://iotmach.utmachala.edu.ec/. (Accessed on 05/02/2018).

[93] Raspberry pi zero w - raspberry pi. https://www.raspberrypi.org/products/raspberry-pi-zero-w/. (Accessed on 05/09/2018).

[94] Waspmote mote runner - 6lowpan development platform - ipv6 for the internet of things and sensorswaspmote mote runner - 6lowpan development platform - ipv6 for the internet of things and sensors — libelium. http://www.libelium.com/products/waspmote-mote-runner-6lowpan/. (Accessed on 05/09/2018).

[95] Specification sheet for the fipy five network development board. https://pycom.io/hardware/fipy_specs/. (Accessed on 05/09/2018).

[96] What is airmax]. https://dl.ubnt.com/AirMax_ppt.pdf. (Accessed on 05/12/2018).

[97] Ubiquiti networks - nanostation® m. https://www.ubnt.com/airmax/nanostationm/. (Accessed on 05/12/2018).

[98] Eclipse mosquitto. https://mosquitto.org/. (Accessed on 05/13/2018).

[99] Mqtt-sn_spec_v1.2.pdf. http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf. (Accessed on 05/13/2018).

[100] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[101] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4):349–359, 2014.

[102] Chayan Sarkar, SN Akshay Uttama Nambi, R Venkatesha Prasad, and Abdur Rahim. A scalable distributed architecture towards unifying iot applications. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 508–513. IEEE, 2014.

[103] Ibrahim Mat, Mohamed Rawidean Mohd Kassim, and Ahmad Nizar Harun. Precision irrigation performance measurement using wireless sensor network. In *Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on*, pages 154–157. IEEE, 2014.

[104] MR Bendre, RC Thool, and VR Thool. Big data in precision agriculture: Weather forecasting for future farming. In *Next Generation Computing Technologies (NGCT), 2015 1st International Conference on*, pages 744–750. IEEE, 2015.

[105] Qin Zhang. *Precision agriculture technology for crop farming*. CRC Press, 2015.

[106] Stepan Ivanov, Kriti Bhargava, and William Donnelly. Precision farming: Sensor analytics. *IEEE Intelligent systems*, 30(4):76–80, 2015.

[107] Karishma Agarwal and Deepak Sharma. Wireless communication wibree (bluetooth low energy technology). *EEC J.*, 2(2):1–4, 2017.

[108] Working groups & committees — bluetooth technology website. https://www.bluetooth.com/specifications/working-groups/working-groups-committees. (Accessed on 04/30/2018).

[109] Gartner says worldwide smartphone sales recorded slowest growth rate since 2013. https://www.gartner.com/newsroom/id/3115517. (Accessed on 05/01/2018).

[110] DK Sreekantha and AM Kavya. Agricultural crop monitoring using iot-a study. In *Intelligent Systems and Control (ISCO), 2017 11th International Conference on*, pages 134–139. IEEE, 2017.

[111] Md Eshrat E Alahi, Li Xie, Subhas Mukhopadhyay, and Lucy Burkitt. A temperature compensated smart nitrate-sensor for agricultural industry. *IEEE Transactions on Industrial Electronics*, 64(9):7333–7341, 2017.

[112] Shailaja M Gunda Nikkam and VR Pawar. Water parameter analysis for industrial application using iot. In *Applied and Theoretical Computing and Communication Technology (iCATccT), 2016 2nd International Conference on*, pages 703–707. IEEE, 2016.

[113] Sunil Kumar Maddikatla and Srivatsava Jandhyala. An accurate all cmos temperature sensor for iot applications. In *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, pages 349–354. IEEE, 2016.

[114] Luke Russell, Rafik Goubran, Felix Kwamena, and Frank Knoefel. Sensor modality shifting in iot deployment: measuring non-temperature data using temperature sensors. In *Sensors Applications Symposium (SAS), 2017 IEEE*, pages 1–6. IEEE, 2017.

[115] Ian Sinclair. *Sensors and transducers*. Elsevier, 2000.

[116] Sam Lucero. IoT platforms: enabling the internet of things. https://cdn.ihs.com/www/pdf/enabling-IOT.pdf, mar 2016. (Accessed on 04/30/2018).

[117] Dominique D Guinard and VM Trifa. Using the web to build the iot. 2016.

[118] Rafael Aguilar, Sergio; Vidal and Carles Gomez. Opportunistic sensor data collection with bluetooth low energy. *Sensors*, 17(1):159, 2017.

[119] Core specifications — bluetooth technology website. https://www.bluetooth.com/specifications/bluetooth-core-specification. (Accessed on 06/05/2018).

[120] Proximity.directory — your guide to proximity marketing. https://www.proximity.directory/. (Accessed on 06/05/2018).

[121] Proximity Technologies in Smart Cities. The proximity.directory report state of the proximity industry. Technical report, Technical Report; Proximity.Directory: Q4, 2016.

[122] Abi Research:. , title = Bluetooth Smart Evolution Helps the Technology Break into Key IoT Market Verticals, url = https://www.abiresearch.com/press/bluetooth-smart-evolution-helps-technology-break-k/ (accessed on 20th November 2017), Available online.

[123] Google Beacon Platform: Eddystone format. , url = https://developers.google.com/beacons/eddystone/ (accessed on 20th November 2017), Available online.

[124] Dixys L Hernández-Rojas, Tiago M Fernández-Caramés, Paula Fraga-Lamas, and Carlos J Escudero. Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing through BLE beacons in IoT telemetry applications. *Sensors*, 18(1), December 2017.

[125] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The internet of things for health care: a comprehensive survey. *IEEE Access*, 3:678–708, 2015.

[126] L. Catarinucci, R. Colella, A. Esposito, L. Tarricone, and M. A Zappatore. Context-aware smart infrastructure based on rfid sensor-tags and its applications to the health-care domain. In *Proceedings of the 2009 IEEE Conference on Emerging Technologies & Factory Automation (ETFA2009)*, pages 22–26, Spain, pp. 1–8, September 2009. Mallorca.

[127] Gonzalo Cerruela García, Irene Luque Ruiz, and Miguel Ángel Gómez-Nieto. State of the art, trends and future of bluetooth low energy, near field communication and visible light communication in the development of smart cities. *Sensors*, 16(11):1968, 2016.

[128] Paula Fraga-Lamas, Tiago M Fernández-Caramés, Manuel Suárez-Albela, Luis Castedo, and Miguel González-López. A review on internet of things for defense and public safety. *Sensors*, 16(10):1644, 2016.

[129] P. Fraga-Lamas, L. Castedo-Ribas, A. Morales-Méndez, and J. M. Camas-Albar. Evolving military broadband wireless communication systems: Wimax. In In Proceedings, editor, *LTE and WLAN*, pages 23–24, Brussels, Belgium, pp. 1–8, May 2016. of the International Conference on Military Communications and Information Systems (ICMCIS).

[130] Tiago M. Fernández-Caramés, Paula Fraga-Lamas, M. Suárez-Albela, and L. Castedo. *A Methodology for Evaluating Security in Commercial RFID Systems, Radio Frequency Identification*. Rijeka, Croatia, 2016, In *Radio Frequency Identification*, 1st ed.; Crepaldi, P. C., Pimenta, T. C., Eds.; INTECH, 2017.

[131] Santiago Barro-Torres, Tiago M Fernández-Caramés, Héctor J Pérez-Iglesias, and Carlos J Escudero. Real-time personal protective equipment monitoring system. *Computer Communications*, 36(1):42–50, 2012.

[132] Tiago M Fernández-Caramés, Paula Fraga-Lamas, Manuel Suárez-Albela, and Luis Castedo. Reverse engineering and security evaluation of commercial tags for rfid-based iot applications. *Sensors*, 17(1):28, 2016.

[133] Josman P Pérez-Expósito, Tiago M Fernández-Caramés, Paula Fraga-Lamas, and Luis Castedo. Vinesens: An eco-smart decision-support viticulture system. *Sensors*, 17(3):465, 2017.

[134] T. Schulz, F. Golatowski, and D. Timmermann. Secure privacy preserving information beacons for public transportation systems. In N. S. W. Sydney, editor, *Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6, 2016.

[135] Paula Fraga-Lamas, Tiago M Fernández-Caramés, and Luis Castedo. Towards the internet of smart trains: a review on industrial iot-connected railways. *Sensors*, 17(6):1457, 2017.

[136] S. J. Barro-Torres, T. M. Fernández-Caramés, M. González-López, and C. J. Escudero-Cascón. Maritime freight container management system using rfid. In La Manga del Mar Menor and, editor, *Proceedings of the Third International EURASIP Workshop on RFID Technology*. Spain, September 2010.

[137] Paula Fraga-Lamas, Diego Noceda-Davila, Tiago M Fernández-Caramés, Manuel A Díaz-Bouza, and Miguel Vilar-Montesinos. Smart pipe system for a shipyard 4.0. *Sensors*, 16(12):2186, 2016.

[138] P. Fraga-Lamas, T. M. Fernández-Caramés, D. Noceda-Davila, and M. Rss Vilar-Montesinos. Stabilization techniques for a real-time passive uhf rfid pipe monitoring system for smart shipyards. In Ieee Rfid, editor, *Proceedings of the 2017 IEEE International Conference on RFID*, pages 9–11, AZ, USA, May 2017. Phoenix.

[139] P. Fraga-Lamas and T. M. Fernández-Caramés. In *Proceedings of the 2017 IEEE International Conference on RFID , editor = Ieee Rfid, month = May, pages = 9-11, publisher = , title = Reverse Engineering the Communications Protocol of an RFID Public Transportation Card, year = 2017,*, Phoenix, AZ, USA, pp. 30–35.

[140] Manuel Suárez-Albela, Tiago M Fernández-Caramés, Paula Fraga-Lamas, and Luis Castedo. A practical evaluation of a high-security energy-efficient gateway for iot fog computing applications. *Sensors*, 17(9):1978, 2017.

[141] Suranga Seneviratne, Yining Hu, Tham Nguyen, Guohao Lan, Sara Khalifa, Kanchana Thilakarathna, Mahbub Hassan, and Aruna Seneviratne. A survey of wearable devices and challenges. *IEEE Communications Surveys & Tutorials*, 19(4):2573–2620, 2017.

[142] Kyriakos Georgiou, Samuel Xavier-de Souza, and Kerstin Eder. The iot energy challenge: A software perspective. *IEEE Embedded Systems Letters*, 2017.

[143] Waleed Ejaz, Muhammad Naeem, Adnan Shahid, Alagan Anpalagan, and Minho Jo. Efficient energy management for the internet of things in smart cities. *IEEE Communications Magazine*, 55(1):84–91, 2017.

[144] Tiago M Fernández-Caramés. An intelligent power outlet system for the smart home of the internet of things. *International Journal of Distributed Sensor Networks*, 11(11):214805, 2015.

[145] Manuel Suárez-Albela, Paula Fraga-Lamas, Tiago M Fernández-Caramés, Adriana Dapena, and Miguel González-López. Home automation system based on intelligent transducer enablers. *Sensors*, 16(10):1595, 2016.

[146] O. Blanco-Novoa, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo. An electricity-price aware open-source smart socket for the internet of energy. *Sensors*, 17:643, 2017.

[147] A. K. Das, S. Zeadally, and M. Wazid. Lightweight authentication protocols for wearable devices. *Computers & Electrical Engineering*, 63:196–208, 2017.

[148] Pallavi Sethi and Smruti R Sarangi. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017, 2017.

[149] C. H. Chen, M. Y. Lin, and W. H. Designing and Lin. and implementing a lightweight wsn mac protocol for smart home networking applications. *Journal of Circuits, Systems and Computers*, 26, 2017.

[150] Rfc 7252 - the constrained application protocol (coap). https://tools.ietf.org/html/rfc7252. (Accessed on 04/30/2018).

[151] Ipv6 over low power wpan (6lowpan) -. https://datatracker.ietf.org/wg/6lowpan/charter/. (Accessed on 04/30/2018).

[152] A. P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby, and M. Architecture and Zorzi. Architecture and protocols for the internet of things: A case study. In Percom Workshops, editor, *Proceedings of the 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 678–683, Germany, 29 March-2, April 2010. Mannheim.

[153] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Lithe: Voigt. Lightweight secure coap for the internet of things. *IEEE Sensors Journal*, 13:3711–3720, 2013.

[154] Rfc 6347 - datagram transport layer security version 1.2. https://tools.ietf.org/html/rfc6347. (Accessed on 06/05/2018).

[155] N. Li, D. Liu, and S. Nepal. Lightweight mutual authentication for iot and its applications. *IEEE Transactions on Sustainable Computing*, 2:359–370, 2017.

[156] M. A. Jan, P. Nanda, X. He, Z. Tan, and R. P. A Liu. Robust authentication scheme for observing resources in the internet of things environment. In *Proceedings of the 2014 IEEE 13th International Conference on Trust*, pages 24–26, Beijing, China, pp. 205-211, September 2014. Security and Privacy in Computing and Communications.

[157] J. Y. Lee, W. C. Lin, and Y. H. A Huang. A lightweight authentication protocol for internet of things. In *Proceedings of the 2014 International Symposium on Next-Generation Electronics (ISNE)*, pages 7–10, Taiwan, pp. 1-2, May 2014. Kwei-Shan.

[158] S. Koteshwara and A. Das. Comparative study of authenticated encryption targeting lightweight iot applications. *IEEE Design & Test*, 34:26–33, 2017.

[159] M. T. Arafin, M. Gao, and G. VOLtA: Qu. Voltage over-scaling based lightweight authentication for iot applications. *In Proceedings of the*, 2017(22):336–341, January 2017.

[160] D. K. Choi, J. H. Jung, H. W. Kang, and S. J. Koh. Cluster-based coap for message queueing in intemet-of-things networks. In *Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 19–22, PyeongChang, Korea, pp. 584-588, February 2017. Phoenix Park.

[161] J. I. Choi and H. S. Yong. Implementation of close interval indoor positioning using beacon. *Navigation*, 12:34–38, 2017.

[162] J. Green and B. Otterdahl. Resource-constrained industrial things-proposal for the adaptation of coap to ethernet/ip™. *In Proceedings of the*, 2017:1–15, February 2017.

[163] W. T. Cheah and C. F. Liao. On findability issues of constrained web of things in a smart home environment. In *Proceedings of the 2017 International Conference on Platform Technology and Service (PlatCon)*, pages 13–15, Korea, pp. 1-6, February 2017. Busan.

[164] Web thing model. https://www.w3.org/Submission/wot-model/. (Accessed on 06/05/2018).

[165] G. Klas, F. Rodermund, Z. Shelby, S. Akhouri, and J. Höller. *Lightweight M2M: Enabling Device Management and Applications for the Internet of Things*. San Diego, CA, USA, ; White paper; Open Mobile Alliance (OMA), 2014.

[166] S. Rao, D. Chendanda, C. Deshpande, and V. Lakkundi. Implementing lwm2m in constrained iot devices. In *Proceedings of the 2015 IEEE Conference on Wireless Sensors (ICWiSe)*, pages 24–26, Malaysia, pp. 52-57, August 2015. Melaka.

[167] Bluetooth technology website. https://www.bluetooth.com/. (Accessed on 04/30/2018).

[168] P. Fraga-Lamas, Noceda-Davila Fernández-Caramés, D.; Díaz-Bouza, M.; Vilar-Montesinos, J. D. M.; Pena-Agras, and L. Castedo. Enabling automatic event detection for the pipe workshop of the shipyard 4.0. *In Proceedings of the 3th FITCE Congress, Madrid, Spain, 14-16*, 2017(56):20–27, September 2017.

[169] J. Wang, E. Schlunt, B. Otis, and T. A Deyle. new vision for smart objects and the internet of things: Mobile robots and long-range uhf rfid sensor tags. *CoRR 2015*, pages 1–8, 2015.

[170] R. Dominikus and J.-m. Schmidt. *Connecting Passive RFID Tags to the Internet of Things*. In Proceedings of the Interconnecting Smart Objects with the Internet Workshop, Prague, Czech Republic, 25, 2011.

[171] F. Nekoogar and F. Dowla. Passive rfid for iot using uwb/uhf hybrid signaling. *In Proceedings of the IEEE Int on Wireless Information Technology and Systems (ICWITS) and Applied Computational Electromagnetics (ACES), Honolulu, HI, USA*, pages 13–17, March 2016.

[172] R. Davidson, K. Townsend, C. Wang, and C. Cufí. Getting started with bluetooth low energy. tools and techniques for low-power networking. 1, 2014.

[173] Silicon Labs. Developing beacons with bluetooth low energy (ble) technology. ; Technical Report; Silicon Labs: Austin, Texas, United States, 2016.

[174] ibeacon - apple developer. https://developer.apple.com/ibeacon/. (Accessed on 04/30/2018).

[175] Altbeacon - the open proximity beacon. http://altbeacon.org/. (Accessed on 04/30/2018).

[176] Series 10 beacon – the gimbal store. https://store.gimbal.com/collections/beacons/products/s10. (Accessed on 06/05/2018).

[177] Indoor mobile location platform — swirl. http://www.swirl.com/products/. (Accessed on 06/05/2018).

[178] Best in-store analytics & retail engagement program — shopkick. https://www.shopkick.com/platform. (Accessed on 06/05/2018).

[179] Statler S. Beacon Technologies:. *The Hitchhiker's Guide to the Beacosystem*. Apress;, 2016.

[180] Bluetooth beacon developement and tutorials — 2016 oveview of resources and solutions. https://www.postscapes.com/bluetooth-beacon-handbook/. (Accessed on 06/05/2018).

[181] The hitchhikers guide to ibeacon hardware: A comprehensive report by aislelabs (2015) - aislelabs. https://www.aislelabs.com/reports/beacon-guide/. (Accessed on 06/05/2018).

[182] Bluetooth beacons market size & share — industry report, 2018-2025. https://www.grandviewresearch.com/industry-analysis/bluetooth-beacons-market/methodology. (Accessed on 06/05/2018).

[183] D. Deugo. Using beacons for attendance tracking. In *Proceedings of the 12th International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'16)*, pages 25–28, Nevada, USA, July 2016. Las Vegas.

[184] Us9374666b1 - beacon communication system and methods - google patents. https://patents.google.com/patent/US9374666. (Accessed on 06/05/2018).

[185] VS Srinivasan, Tnr Kumar, Dushyanth Kumar Yasarapu, et al. Raspberry pi and ibeacons as environmental data monitors and the potential applications in a growing bigdata ecosystem. pages 961–965, 2016.

[186] P. C. Deepesh, R. Rath, A. Tiwary, V. N. Rao, and N. Kanakalata. Experiences with using ibeacons for indoor positioning. *In Proceedings of the*, 9:184–189, February 2016.

[187] C. Park, J. Lim, J. Kim, S. J. Lee, and D. Don't Bother Me. I'm Socializing!: A Lee. Breakpoint-based smartphone notification system. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 541–554, Oregon, USA, 25 February - 1, March 2017. Portland.

[188] M. E. Kiziroglou, D. E. Boyle, E. M. Yeatman, and J. J. Cilliers. Opportunities for sensing systems in mining. *IEEE Transactions on Industrial Informatics*, 13:278–286, 2017.

[189] S. Rajpoot, S. Kumar, and P. Singh. Implementing the physical web using bluetooth low energy based beacons and a mobile app. In *Proceedings of the 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, pages 3–5, India, pp. 327–329, February 2016. Noida.

[190] Debasis Bhattacharya, Mario Canul, and Saxon Knight. Impact of the physical web and ble beacons. In *Proceedings of the 5th Annual Conference on Research in Information Technology*, pages 53–53. ACM, 2016.

[191] Jennifer BONACINA. Proximity marketing and its technologies: over 100 beacon use cases analysis. msc's degree . università degli studio di bergamo. dipartimento di scienze aziendali. 2016.

[192] Alexandre Alapetite and John Paulin Hansen. Dynamic bluetooth beacons for people with disabilities. pages 36–41, 2016.

[193] Wo2016060696a1 - interleaving multiple bluetooth low energy advertisements - google patents. https://patents.google.com/patent/WO2016060696A1/en. (Accessed on 06/05/2018).

[194] Us20160050564a1 - bluetooth transmission security pattern - google patents. https://patents.google.com/patent/US20160050564. (Accessed on 06/05/2018).

[195] Ariel M Campoverde, Dixys L Hernández R, and Bertha E Mazón O. Cloud computing con herramientas open-source para internet de las cosas. *Maskana*, 6(Supl.):173–182, December 2015.

[196] Fumio Narita and Marina Fox. A review on piezoelectric, magnetostrictive, and magnetoelectric materials and device technologies for energy harvesting applications. *Advanced Engineering Materials*, 20(5):1700743, 2018.

[197] Dominique Guinard and Vlad Trifa. *Using the Web to Build the IoT*. Manning Publications, Shelter Island, NY, USA, 2016.

[198] Dixys L Hernández-Rojas, Tiago M Fernández-Caramés, Paula Fraga-Lamas, and Carlos J Escudero. A plug-and-play human-centered virtual teds architecture for the web of things. *Sensors*, 18(7), June 2018.

[199] Paula Fraga Lamas, Tiago M Fernández Caramés, Ángel Carro Lagoa, Carlos J Escudero Cascón, and Miguel González López. Estándares para interoperabilidad de redes de sensores: IEEE 1451 y sensor web enablement (SWE). January 2014.

[200] Kang Lee. IEEE 1451 and IEEE 1588 standards. https://www.nist.gov/sites/default/files/documents/el/isd/ieee/Information-on-1451_1588-V36.pdf, 2008. (Accessed on 04/30/2018).

[201] Mikhaylov, K., Jämsä, J., Luimula, M., Tervonen, J. Intelligent sensor interface and data format. In *In Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning*, pages 55–76. CRC Press.

[202] Sensordata-buster. Sensor plug & play the new standard for automated sensor measurement. http://www.sensordata-burster.com/wp-content/uploads/bsk-pdf-manager/78_SENSORDATA_PRODUCTS_WITH_TEDS.PDF, 2006. (Accessed on 04/30/2018).

[203] E Song and K Lee. Smart transducer web services based on the IEEE 1451.0 standard. In *2007 IEEE Instrumentation Measurement Technology Conference IMTC 2007*, pages 1–6, May 2007.

[204] Song, E. Y., & Lee, K. B. STWS: A unified web service for IEEE 1451 smart transducers. *IEEE Transactions on Instrumentation and Measurement*, 57(8):1749–1756, 2008.

[205] Eugene Y Song and Lee Kang. Integration of IEEE 1451 smart transducers and OGC-SWE using STWS. In *2009 IEEE Sensors Applications Symposium*, 2009.

[206] Shadi Abou-Zahra, Jeff Jaffe, and Marie-Claire Forgue. World wide web consortium (w3c). https://www.w3.org/. (Accessed on 04/30/2018).

[207] Open Geospatial Consortium Inc., C. Bruce (Ed.). Binary extensible markup language (BXML) encoding specification (OGC 03-002r9). http://portal.opengeospatial.org/files/?artifact_id=13636, 2006. (Accessed on 04/30/2018).

[208] Iso/iec 24824-1:2007 — iec webstore. https://webstore.iec.ch/publication/11239. (Accessed on 04/30/2018).

[209] Schneider And Kamiya. Efficient xml interchange (exi) format 1.0 (second edition). https://www.w3.org/TR/exi/, 2011. (Accessed on 04/30/2018).

[210] Torben R Licht. The IEEE 1451.4 proposed standard and emerging compatible smart transducers and systems. https://standards.ieee.org/develop/regauth/tut/1451d4.pdf, 2000. (Accessed on 04/30/2018).

[211] Nenad Jevtic and Vujo Drndarevic. Design and implementation of Plug-And-Play analog resistance temperature sensor. *Metrology and Measurement Systems*, 20(4)(565-580), December 2013.

[212] J Higuera, W Hertog, M Perálvarez, J Polo, and J Carreras. Smart lighting system ISO/IEC/IEEE 21451 compatible. *IEEE Sens. J.*, 15(5):2595–2602, May 2015.

[213] A Kumar, V Srivastava, M K Singh, and G P Hancke. Current status of the IEEE 1451 Standard-Based sensor applications. *IEEE Sens. J.*, 15(5):2505–2513, May 2015.

[214] G Corotinschi and V G Găitan. The development of IoT applications using old hardware equipment and virtual TEDS. In *In Proceedings of the 2016 International Conference on Development and Application Systems (DAS)*, pages 264–268, Suceava, Romania, 2016.

[215] Z Liu, G Monte, and V Huang. ISO/IEC/IEEE P21451-001 standard for signal treatment of sensory data. In *In Proceedings of the 2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, pages 766–771, Santa Clara, USA, 2016.

[216] K S E Phala, A Kumar, and G P Hancke. Air quality monitoring system based on ISO/IEC/IEEE 21451 standards. *IEEE Sens. J.*, 16(12):5037–5045, June 2016.

[217] Paul Celicourt and Michael Piasecki. An IEEE 1451.0-based Platform-Independent TEDS creator using open source software components. *Int. J. Sensors Sensor Netw*, 3(1)(1-11):1–11, March 2015.

[218] O S Ajigboye and K Danas. Towards semantics in wearable sensors: The role of transducers electronic data sheets (TEDS) ontology in sensor networks. In *In Proceedings of the 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6, Munich, Germany, 2016.

[219] Fangling Pu, Zhili Wang, Chong Du, Wenchao Zhang, and Nengcheng Chen. Semantic integration of wireless sensor networks into open geospatial consortium sensor observation service to access and share environmental monitoring systems. *IET Softw.*, 10(2):45–53, April 2016.

[220] Manuel Suárez-Albela, Paula Fraga-Lamas, Tiago M Fernández-Caramés, Adriana Dapena, and Miguel González-López. Home automation system based on intelligent transducer enablers. *Sensors*, 16(10), September 2016.

[221] Sensor web enablement (swe) — ogc. http://www.opengeospatial.org/ogc/markets-technologies/swe. (Accessed on 04/30/2018).

[222] Carl Reed, Mike Botts, and John Davidson Ogc. OGC® sensor web enablement: Overview and high level architecture. In *In Proceedings of the IEEE Autotestcon*, number 3, pages 372–380, Baltimore, USA, 2007.

[223] Michel Grothe, Jan Kooijman, and Nederlandse Commissie voor Geodesie. *Sensor web enablement*, volume 45. NCG Nederlandse Commissie voor Geodesie, 2008.

[224] S Jirka, D Nüst, J Schulte, and F Houbie. Integrating the OGC sensor web enablement framework into the OGC catalogue. In *In Proceedings of the WebMGS*, pages 26–27, Como, Italy, 2010.

[225] Bröring, A., Janowicz, K., Stasch, C., & Kuhn, W. Semantic challenges for sensor plug & play. In *In Proceedings of the Web and Wireless Geographical Information Systems Springer Berlin Heidelberg.*, pages 72–86, Berlin, Germany, 2009.

[226] David Díaz Pardo De Vera, Alvaro Sigüenza Izquierdo, Jesús Bernat Vercher, and Luis Alfonso Hernández Gómez. A ubiquitous sensor network platform for integrating smart devices into the semantic sensor web. *Sensors*, 14(6):10725–10752, June 2014.

[227] Lissette Muñoz Guillén, José Medina Cartuche, Juan Guillermo Aungüisaca, and Jaime Veintimilla Reyes. Arquitectura para una red de sensores web basada en SWE (sensor web enablement): Caso de estudio para la implementación en sensores hidrometeorológicos. *Maskana*, 7(Supl.):129–138, January 2017.

[228] Semantic sensor network ontology. https://www.w3.org/2005/Incubator/ssn/ssnx/ssn. (Accessed on 04/30/2018).

[229] Chih-Yuan Huang and Steve Liang. AHS model: Efficient topological operators for a sensor web Publish/Subscribe system. *ISPRS International Journal of Geo-Information*, 6(2):54, February 2017.

[230] Mike Botts. Sensor model language (sensorml) — ogc. http://www.opengeospatial.org/standards/sensorml, 2007.

[231] Tom O'Reilly. Ogc®puck protocol standard — ogc. http://www.opengeospatial.org/standards/puck, 2012. (accessed on June 2018).

[232] Ogc puck standard enables 'plug and work' sensor networks — ogc. http://www.opengeospatial.org/pressroom/pressreleases/1542, 2012. (accessed on June 2018).

[233] Daniel Mihai Toma, Thomas C O'Reilly, Arne Bröring, David R Dana, Felix Bache, Kent L Headley, Antoni Mànuel-Làzaro, Duane R Edgington, et al. Standards-based plug & work for instruments in ocean observing systems. *IEEE Journal of Oceanic Engineering*, 39(3):430–443, 2014.

[234] Mikhaylov, K., Petäjäjärvi, J., Mäkeläinen, M., Paatelma, A., & Hänninen, T. Extensible modular wireless sensor and actuator network and IoT platform with Plug&Play module connection. In *In Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 386–387, Seattle, USA, 2015.

[235] Nelson Matthys, Fan Yang, Wilfried Daniels, Thomas Watteyne Sam Michiels, Wouter Joosen, and Danny Hughes. PnP-Mesh: the Plug-and-Play mesh network for the internet of things. In *In Proceedings of the IEEE World Forum on Internet of Things*, pages 311–315, Milan, Italy, 2015.

[236] Sohail Jabbar, Farhan Ullah, Shehzad Khalid, Murad Khan, and Kijun Han. Semantic interoperability in heterogeneous IoT infrastructure for healthcare. *Proc. Int. Wirel. Commun. Mob. Comput. Conf.*, 2017, March 2017.

[237] Rdf 1.1 primer. https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/. (Accessed on 04/30/2018).

[238] Arne Bröring, Stefan Schmid, Corina-Kim Schindhelm, Abdelmajid Khelil, Sebastian Kaebisch, Denis Kramer, Danh Le Phuoc, Jelena Mitic, Darko Anicic, Ernest Teniente, and Others. Enabling IoT ecosystems through platform interoperability. http://www.arne-broering.de/BIG%20IoT%20-%20Vision.pdf, 2017. (Accessed on 04/30/2018).

[239] Big iot – bridging the interoperability gap of the internet of things. http://big-iot.eu/. (Accessed on 04/30/2018).

[240] Niteen Mohod. Usability of internet of things [IoT] for dam safety and water management. *International Journal of Research in Advent Technology*, 5(1):29–30, January 2017.

[241] Mukaddim Pathan, Kerry Taylor, and Michael Compton. Semantics-based Plug-and-Play configuration of sensor network services. In *In Proceedings of the 3rd International Conference on Semantic Sensor Networks-Volume 668*, pages 17–32, Shanghai, China, November 2010.

[242] Amelie Gyrard, Martin Serrano, and Pankesh Patel. Building interoperable and cross-domain semantic web of things applications. In *Managing the Web of Things*, pages 305–324. Elsevier, 2017.

[243] Pankesh Patel, Amelie Gyrard, Dhavalkumar Thakker, Amit Sheth, and Martin Serrano. Swotsuite: A development framework for prototyping cross-domain semantic web of things applications. *arXiv preprint arXiv:1609.09014*, 2016.

[244] Fiesta-iot - federated interoperable semantic iot testbeds and applications. http://fiesta-iot.eu/. (Accessed on 04/30/2018).

[245] Openiot – open source cloud solution for the internet of things. http://www.openiot.eu/. (Accessed on 04/30/2018).

[246] Arne Bröring, Felix Bache, Thomas Bartoschek, and Corné PJM van Elzakker. The sid creator: A visual approach for integrating sensors with the sensor web. In *Advancing Geoinformation Science for a Changing World*, pages 143–162. Springer, 2011.

[247] Daniel Mihai Toma. Technology transfer of observatory software. https://www3.mbari.org/pw/2010-TTOSInternProject-v1.pdf, 2010. (accessed on June 2018).

[248] M. Al-Soh and I.A. Zualkernan. An mqtt-based context-aware wearable assessment platform for smart watches. In *Proceedings of the IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, pages 98–100. Timisoara, July 2017.

[249] S. Ahmed, A. Topalov, and N. Shakev. A robotized wireless sensor network based on mqtt cloud computing. In *Proceedings of the 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pages 1–6. Donostia-San Sebastian, Spain, May. 2017.

[250] A. Sinha, S. Sharma, P. Goswami, V. K. Verma, and M. Manas. Design of an energy efficient iot enabled smart system based on dali network over mqtt protocol. In *Proceedings of the 3rd International Conference on Computational Intelligence and Communication Technology (CICT)*, pages 1–5. Ghaziabad, Feb. 2017.

[251] B. Oryema, H. S. Kim, W. Li, and J. T. Park. Design and implementation of an interoperable messaging system for iot healthcare services. In *Proceedings of the 14th IEEE Annual Consumer Communications and Networking Conference (CCNC)*, pages 45–52. Las Vegas, NV, Jan. 2017.

[252] Bill Earl. calibrating-sensors. https://cdn-learn.adafruit.com/downloads/pdf/calibrating-sensors.pdf, 2016. (accessed on June 2018).

[253] Dixys Hernandez-Rojas and others. IoT android gateway for monitoring and control a WSN. *International Conference on Technology Trends. Springer*, pages 18–32, November 2017.

[254] Conjunto de aplicaciones de iot de azure — microsoft azure. https://azure.microsoft.com/es-es/suites/iot-suite/. (Accessed on 05/02/2018).

[255] Google cloud platform pricing calculator — google cloud platform — google cloud. https://cloud.google.com/products/calculator/?hl=es#id=8bcb41f2-8eb1-4baf-8b86-d003aef848a4. (accessed on June 2018).

[256] Android studio and sdk tools — android developers. Available online: https://developer.android.com/studio/?hl=es-419. (accessed on May 2018).

[257] Serial port monitor - rs232 port sniffer & analyzer - serial monitor. Available online: https://www.eltima.com/products/serial-port-monitor/. (accessed on May 2018).

[258] Products — navicat. Available online: https://www.navicat.com/en/products. (accessed on May 2018).

[259] Tera term official website. Available online: https://ttssh2.osdn.jp/index.html.en. (accessed on May 2018).

[260] nrf51422 official website. Available online: https://www.nordicsemi.com/eng/Products/ANT/nRF51422. (accessed on May 2018).

[261] Cortex-m0 official arm website. Available online: https://developer.arm.com/products/processors/cortex-m/cortex-m0. (accessed on May 2018).

[262] Atmega2560 official microship's website. Available online: http://www.microchip.com/wwwproducts/en/ATmega2560. (accessed on May 2018).

[263] nrf52840 / products / home - ultra low power wireless solutions from nordic semiconductor. http://www.nordicsemi.com/eng/Products/nRF52840. (Accessed on 05/21/2018).

[264] Software - pycom. https://pycom.io/software-2/. (Accessed on 05/21/2018).

[265] Atom ide. https://ide.atom.io/. (Accessed on 05/21/2018).

[266] Sublime text - a sophisticated text editor for code, markup and prose. https://www.sublimetext.com/. (Accessed on 05/21/2018).

[267] Visual studio code - code editing. redefined. https://code.visualstudio.com/. (Accessed on 05/21/2018).

[268] Pycharm: Python ide for professional developers by jetbrains. https://www.jetbrains.com/pycharm/. (Accessed on 05/21/2018).

[269] Universidad Técnica de Machala. Proyecto Riego Bananera - Resolución No. 396/2016 H. Consejo Universitario UTMACHALA. https://drive.google.com/drive/u/0/folders/0B0QMiL4rtDtlZmZzdFd0bnRUTzA?ogsrc=32, 2016. (Accessed on 08/01/2018).

[270] The lopy4 is a quadruple bearer micropython enabled development board including lora, sigfox, wifi and bluetooth. https://pycom.io/product/lopy4/. (Accessed on 05/21/2018).

[271] Universidad Técnica de Machala. Resolución No. 390/2017 H. Consejo Universitario UTMACHALA. http://www.utmachala.edu.ec/archivos/filesportal/2016/SECRETARIA/RESOLUCIONES2017/AGOSTO/2017-08-08/RES_390.PDF, 2017. (Accessed on 08/01/2018).

[272] Universidad Técnica de Machala. Resolución No. 607/2016 H. Consejo Universitario UTMACHALA. http://www.utmachala.edu.ec/archivos/filesportal/2016/SECRETARIA/RESOLUCIONES/DICIEMBRE/2016-12-08/607.PDF, 2016. (Accessed on 08/01/2018).

[273] Universidad Técnica de Machala. Resolución. No. 362/2017 H. Consejo Universitario UTMACHALA. http://www.utmachala.edu.ec/archivos/filesportal/2016/SECRETARIA/RESOLUCIONES2017/JULIO/2017-06-25/RES_362.PDF, 2017. (Accessed on 08/01/2018).